Escola Universitaria Politécnica

UNIVERSIDADE DA CORUÑA

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO DE FIN DE GRADO

TFG Nº: **770G01A164**

TÍTULO: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**

AUTOR: **DAVID GÓMEZ OCAMPO**

TUTOR: **ALBERTO JOSÉ LEIRA REJAS**

FECHA: **FEBRERO DE 2020**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

TÍTULO: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**

ÍNDICE

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: FEBRERO DE 2020

AUTOR: EL ALUMNO

Fdo.: DAVID GÓMEZ OCAMPO

Listado de figuras

Listado de tablas

TÍTULO: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**

MEMORIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: FEBRERO DE 2020

AUTOR: EL ALUMNO

Fdo.: DAVID GÓMEZ OCAMPO

Índice del documento MEMORIA

1 OBJETO

Se desarrollará una librería de diez aplicaciones para un robot Lego, en grado creciente de complejidad, con una explicación detallada del código empleado e instrucciones precisas para su montaje, que servirán para un posible uso posterior en la enseñanza o en demostraciones prácticas.

2 ALCANCE

El presente proyecto abordará los siguientes puntos:

- Diseño y construcción de los robots.
- Recreación 3D de los robots mediante herramienta CAD.
- Programación de los robots.

3 ANTECEDENTES

3.1. Lego Mindstorms

Lego Mindstorms es una plataforma de hardware y software fabricada por LEGO, surgida de la colaboración de éstos y el MIT (Massachusetts Institute of Technology) en la década de los 80.

El primer prototipo funcional se creó en 1987, aunque la primera generación de Mindstorms no saldría al mercado hasta 1998.

Hasta la fecha LEGO ha desarrollado tres generaciones de Mindstorms: RCX (lanzado en 1998), NXT (2006) y EV3 (2013), para cada versión se creó un kit educativo que además del bloque programable o “brick” incluía diversos actuadores y sensores.

- RCX:
  El bloque programable de la primera versión podía programarse utilizando el lenguaje RCX o bien RoboLab, una versión adaptada de LabView de National Instruments.

Ambos lenguajes están basados en iconos que permiten crear diagramas de flujo, lo que hace que la programación sea muy visual, ideal para el aprendizaje, esta filosofía se mantendría en las versiones posteriores.

La versión RCX utilizaba un bloque con un microcontrolador de 16MHz y una memoria RAM de 32K, el kit educativo incluía dos motores, dos sensores táctiles y un sensor de luz.

Figura 3.1 – Bloque programable RCX

- NXT:
  El kit educativo NXT contenía tres servomotores, tres sensores de luz, sonido y distancia respectivamente, y un sensor táctil (dos en el caso de la versión NXT 2.0). El bloque funcionaba con un microcontrolador a 48MHz con 64KB de RAM.

  Podía ser programado mediante el entorno gráfico NXT-G, o al igual que la versión anterior, RoboLab.
  Existe software no oficial que nos permite programar en multitud de lenguajes (RobotC, C#, Java, Python. . .) aunque en ocasiones es necesario modificar el firmware del bloque.

Figura 3.2 – Bloque programable NXT

- EV3:
  La versión más reciente hasta la fecha cuenta con un microcontrolador que trabaja a

300MHz, con 64MB de RAM y una memoria Flash de 16MB, bajo un sistema operativo GNU/Linux. Como mejora respecto a NXT, incorpora conectividad WiFi y Bluetooth, además de una ranura para Micro SD.

El kit educativo incluye dos servomotores grandes, un servomotor mediano, dos sensores táctiles, un sensor de color/luz, un sensor giroscópico y un sensor ultrasónico.

El lenguaje de programación 'oficial' para EV3 es de tipo gráfico (EV3-G) y está basado en LabView, aunque de nuevo puede programarse en multitud de lenguajes, siendo popular entre los programadores experimentados el sistema operativo *ev3dev*, que puede cargarse desde una tarjeta MicroSD y permite programar en multitud de lenguajes de alto nivel sin necesidad de modificar el firmware del bloque.

Figura 3.3 – Bloque programable EV3

3.2. Descripción del hardware

3.2.1. Bloque programable (o'brick')

- Sistema operativo LINUX
- Controlador ARM9 a 300MHz
- Memoria Flash 16MB
- RAM 64MB
- 4 puertos de entrada (1,2,3,4), conector RJ12
- 4 puertos de salida (A,B,C,D), conector RJ12
- Resolución de la pantalla 178x128 pixel (Blanco y negro)
- USB 2.0, hasta 480Mbit/segundo (conector miniUSB) para comunicación con PC
- USB 1.1, hasta 12Mbit/segundo, para adaptador Wi-Fi
- Ranura para Micro SD, soporta SDHC, versión 2.0. Máximo 32GB

- 6 pilas AA/batería recargable

Los botones asignables se designan por un número del 1 al 5 del siguiente modo:

Figura 3.4 – Botones del bloque programable EV3

3.2.2. Motor grande

- Encoder (1º de resolución)
- Velocidad max.: 160-170 rpm
- Torque: 20Ncm
- Torque de frenado: 40Ncm

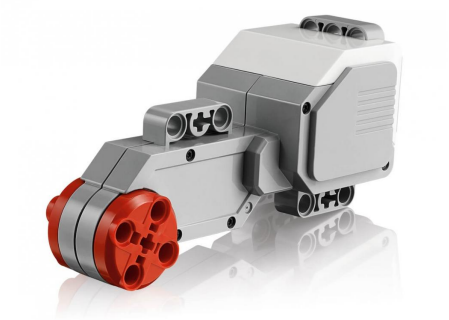


Figura 3.5 – Motor grande EV3

3.2.3. Motor mediano

- Encoder (1º de resolución)
- Velocidad max.: 240-250 rpm
- Torque: 8 Ncm
- Torque de frenado: 12 Ncm

Figura 3.6 – Motor mediano EV3

3.2.4. Sensor de color/luz

- Modo color: reconoce 7 colores (negro, azul, verde, amarillo, rojo, blanco y marrón) o 'sin color', devuelve un valor numérico, dependiendo del caso.

Valor	Color
0	Sin Color
1	Negro
2	Azul
3	Verde
4	Amarillo
5	Rojo
6	Blanco
7	Marrón

Tabla 3.1 – Código de colores

- Modo luz reflejada: Mide la intensidad de luz reflejada, emitida por un diodo led rojo, utiliza una escala de 0 (muy oscuro) a 100 (muy luminoso).
- Modo luz ambiental: Utilizando la misma escala que el modo anterior, mide la intensidad de luz ambiental.

La frecuencia de muestreo del sensor es de 1KHz.

Figura 3.7 – Sensor de color/luz EV3

3.2.5. Sensor giroscópico

Se trata de un sensor digital que detecta movimiento rotatorio (en un solo eje, el que se indica en la carcasa del sensor).
Puede medir la rotación total en grados, con una precisión de +/- 3 grados para un giro de 90º, o la variación de ésta, hasta un máximo de 440º/segundo.

Figura 3.8 – Sensor giroscópico EV3

3.2.6. Sensor táctil

Es simplemente un contacto normalmente abierto, activado por pulsador, por lo que eléctricamente solo tiene dos estados. Sin embargo, a la hora de programar, el software distingue tres estados: pulsado, no pulsado, y pulsación rápida (en inglés 'bumped').

Valor	Estado
0	No Pulsado
1	Pulsado
2	Pulsación rápida

Tabla 3.2 – Sensor táctil: tabla de estados

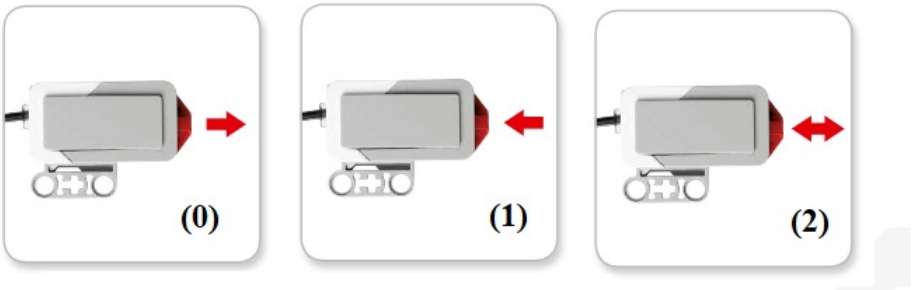


Figura 3.9 – Estados del sensor táctil EV3

3.2.7. Sensor ultrasónico

Se trata de un sensor digital que mide la distancia a objetos emitiendo una señal de alta frecuencia y midiendo el tiempo que tarda la misma en regresar al sensor, dicha distancia puede medirse en cm o pulgadas.

- Rango: 3 a 250 cm, 1 a 99 pulgadas.
- Precisión: +/- 1 cm, +/- 0.394 pulgadas.
- Una lectura de 255 cm o 100 pulgadas, indica que no se detecta ningún objeto.

Permite dos modos de funcionamiento:

- Modo medición: el sensor se ilumina con una luz roja constante.
- Modo presencia: la luz roja es parpadeante, en este modo el sensor no emite, pero detecta las ondas emitidas por otros sensores.

Figura 3.10 – Sensor ultrasónico EV3

3.3. EV3-G

EV3-G es un software basado en LabView, por tanto se basa en la utilización de bloques y su interconexionado. En la parte inferior de la ventana de programación, encontramos 6 pestañas, son las familias o tipos en los que se dividen los bloques.

3.3.1. Elementos de programación: familias de bloques

3.3.1.0.1. Acción (pestaña verde)

- Motor grande y motor mediano: El funcionamiento de estos dos bloques es idéntico. Nos permite elegir la función a realizar:

- Apagado.
- Encendido.
- Encendido por segundos.
- Encendido por rotaciones.
- Encendido por grados.

Nos permite regular la potencia entregada al motor y las rotaciones/grados a girar. Podemos elegir también si queremos que pare el motor tras realizar la acción indicada.

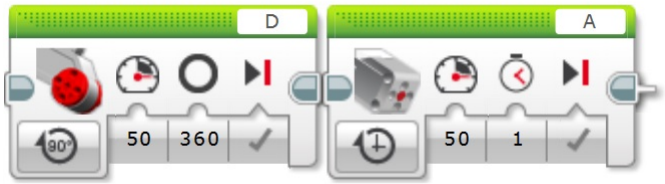


Figura 3.11 – Bloques 'Motor grande' y 'Motor mediano'

En el ejemplo de la figura 3.11, vemos un motor grande conectado al puerto D, configurado para girar 360º y un motor mediano, conectado al puerto A, configurado para girar durante 1s.
En ambos la potencia es de 50 y se detendrán al finalizar la acción.

- Mover la dirección: Este bloque controla dos motores grandes a la vez de manera sincronizada, para ello incorpora el parámetro 'dirección'.
  Un valor de 0, hace que el bloque se desplace en línea recta (ambos motores girando a igual velocidad).
  Un valor positivo hace que el bloque gire hacia la derecha y viceversa.

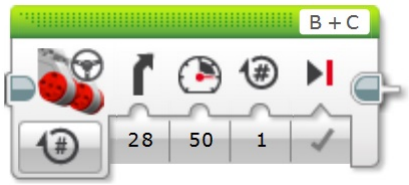


Figura 3.12 – Bloque 'Mover dirección'

- Mover tanque: Funciona como dos bloques individuales, aunque podemos programar giros dando distintos valores de potencia a cada motor, los dos motores no funcionan de manera coordinada como en el bloque anterior, por lo que los giros son menos precisos.

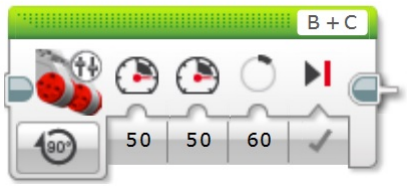


Figura 3.13 – Bloque 'Mover tanque'

- Pantalla: Nos permite mostrar en la pantalla texto, formas geométricas básicas (rectángulo, circulo, punto...) o imágenes predefinidas, además el programa cuenta con un editor que nos permite crear nuestras propias imágenes.
  El parámetro borrar pantalla indica si queremos borrar la pantalla cuando se despliega una nueva imagen, o por el contrario superponerla a la imagen anterior.
  Los parámetros x e y son las coordenadas donde se coloca la imagen.

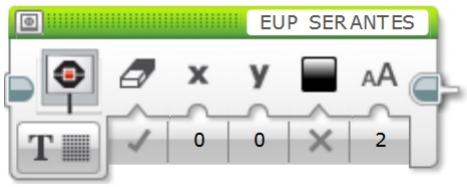


Figura 3.14 – Bloque 'Pantalla'

En el ejemplo 3.14, la pantalla muestra el texto 'EUP SERANTES', con un tamaño de letra 2, centrado en las coordenadas (0,0). El texto se borrará cuando se despliegue otra imagen.

- Sonido: Permite reproducir sonidos predefinidos o bien generados mediante el editor de sonidos del programa.
  En el modo Reproducir Nota se desplegará un teclado musical de tres octavas.
  En el modo Reproducir Tono, nos permite escribir la frecuencia en Hz del tono.
  Podremos seleccionar la duración (en segundos), el volumen (0 – 100) y el tipo de reproducción:

  - Esperar a que se complete [0].
  - Reproducir una vez [1].
  - Reproducir en bucle [2].

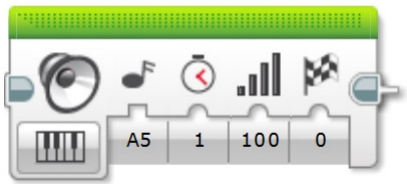


Figura 3.15 – Bloque 'Sonido'

En el ejemplo de la figura 3.15, el bloque emite la nota La durante un segundo, a un volumen de 100, esperando a que termine.

- Luz de estado del bloque EV3: Enciende/apaga la luz de la botonera del bloque de color verde (0), naranja (1) y rojo (2), de manera continua o pulsante.

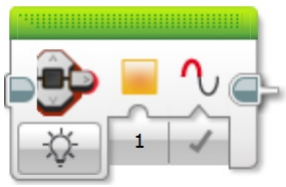


Figura 3.16 – Bloque 'Luz de estado del bloque EV3'

3.3.1.0.2. Control de flujo (pestaña naranja)

- Iniciar: Todos los programas y subprogramas comenzarán necesariamente por este bloque.

Figura 3.17 – Bloque 'Inicio'

- Esperar: Detiene el flujo del programa, un tiempo determinado o bien hasta que uno de los sensores o botones que elijamos tome el valor que le indiquemos.

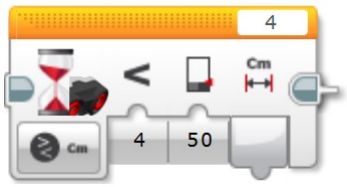


Figura 3.18 – Bloque 'Esperar'

En el ejemplo de la figura 3.18, el bloque detiene el flujo del programa hasta que el sensor ultrasónico conectado al puerto 4, detecta un objeto a una distancia inferior a 50cm.

- Bucle: Ejecuta un bucle de manera ilimitada o un número de veces determinado, también podemos elegir como condición de interrupción de bucle un valor determinado para un sensor o una condición lógica.

  Dispone de una salida que proporciona el número de iteración del bucle que se está ejecutando

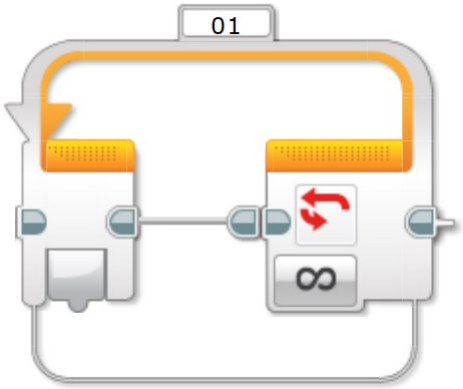


Figura 3.19 – Bucle 'Ilimitado'

- Interruptor: Su nombre proviene de la traducción literal de 'Switch'. Equivale a una estructura tipo «switch» en lenguaje código, ejecuta una acción u otra en función del valor de una variable.

  En otros casos, funciona como una estructura tipo «if. . . else», evalúa la lectura de un sensor o el valor de una variable (numérica, lógica o texto) y lo compara con el valor que le indiquemos, ejecutando una rama del bloque u otra en función de que el resultado de dicha comparación sea verdadero o falso.

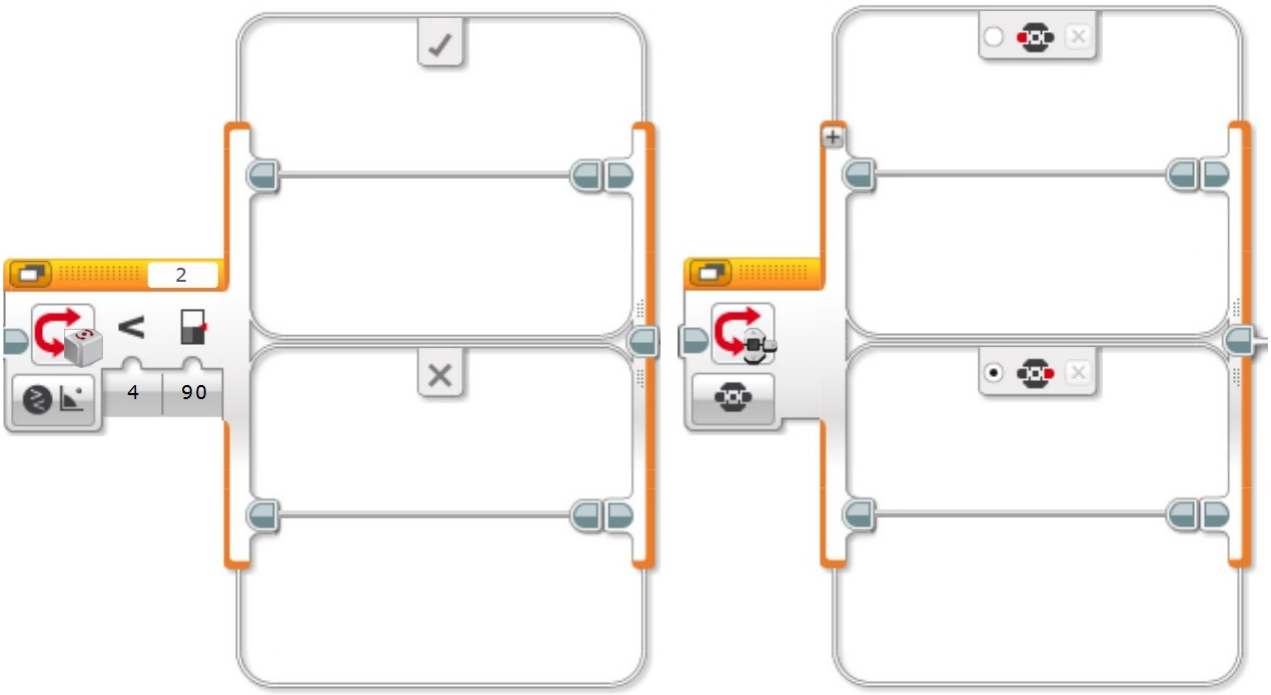


Figura 3.20 – Bloques 'Interruptor'

Figura 3.20: El bloque interruptor de la izquierda funciona como «if... else», evalúa que el ángulo medido por el giroscopio del puerto 2 es menor de 90º (solo hay dos opciones).

El bloque interruptor de la derecha funciona como «switch», ejecutará una línea de código diferente en función del botón que se pulse (tantas opciones como botones).

- Interrupción del bucle: Interrumpe la ejecución del bucle que se le indique en la esquina superior derecha, es útil ya que el bloque 'Bucle' solo evalúa una condición para poder ser interrumpido.

Figura 3.21 – Bloque 'Interrupción del bucle'

3.3.1.0.3. Sensor (pestaña amarilla) Todos los bloques de esta pestaña funcionan de la misma manera y nos permiten obtener la información de los sensores y temporizadores. Disponen de tres funciones:

- Medida: proporciona la lectura del sensor, algunos sensores miden más de una variable y permiten elegir la unidad de medida (por ejemplo: grados o rotaciones).

- Comparar: compara la medida con un valor del modo que le indiquemos según un índice numérico. Proporciona dos salidas, el valor medido y una salida lógica según el resultado de la comparación.

Indice	Comparación
0	Igual
1	Distinto
2	Mayor
3	Mayor o igual
4	Menor
5	Menor o igual

Tabla 3.3 – Comparación: Índices

- Reiniciar: en algunos casos, como en el del sensor giroscópico, los encoders o los temporizadores, nos permite reiniciar la medida.

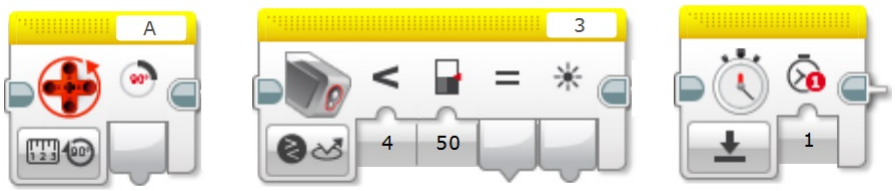


Figura 3.22 – Bloques 'Sensor'

Figura 3.22: Tres bloques 'Sensor' funcionando en modo medida, comparar y reiniciar, respectivamente.

3.3.1.0.4. Operaciones con datos (pestaña roja)

- Variable: nos permite leer/escribir variables de tipo texto, numérico, lógico y arrays de variables de tipo numérico y lógico.
- Constante: Similar al bloque anterior, pero en este caso creamos constantes, es decir no podemos modificar su valor posteriormente.

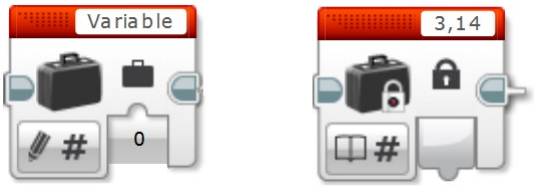


Figura 3.23 – Bloques 'Variable' y 'Constante'

- Operaciones secuenciales: utilizamos este bloque para manipular vectores (de tipo numérico o lógico), dispone de cuatro modos:

  - Anexar: añade un elemento al final del array.
  - Leer en el índice: devuelve el valor del elemento ubicado en el vector en el índice que le indiquemos.
  - Escribir en el índice: modifica el valor de la posición que le indiquemos dentro del vector.
  - Longitud: devuelve el número de elementos del vector.

Figura 3.24 – Bloque 'Operaciones Secuenciales'

El bloque de la figura 3.24 es un bloque de operaciones secuenciales en modo escritura numérico.
Le indicaremos el vector del cual debe leer, el índice de la posición de dicho vector que queremos modificar, el dato a introducir en dicho índice y el vector sobre el cual debe volcar el resultado.

- Operaciones lógicas: bloque configurable como AND, OR, XOR y NOT.

Figura 3.25 – Bloque 'Operaciones lógicas'

- Matemática: permite las operaciones: agregar, sustraer, multiplicar, dividir, valor absoluto, raíz cuadrada y exponencial. Además la función 'avanzado' nos permite introducir una función personalizada, con un máximo de cuatro operandos.

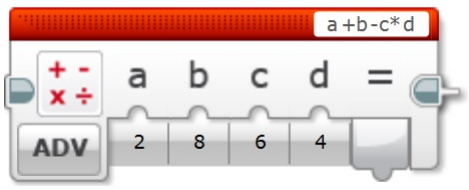


Figura 3.26 – Bloque 'Operaciones matemáticas'

- Redondear: redondea al valor más cercano, hacia arriba o hacia abajo, también nos da la opción de truncar decimales.

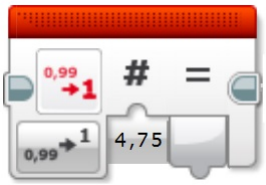


Figura 3.27 – Bloque 'Redondear'

- Comparar: compara valores numéricos del modo que le indiquemos y devuelve un valor lógico.

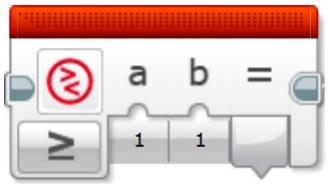


Figura 3.28 – Bloque 'Comparar'

- Rango: devuelve un valor lógico si el valor que le indicamos está dentro o fuera (según la configuración elegida) de los límites superior e inferior que proporcionemos.

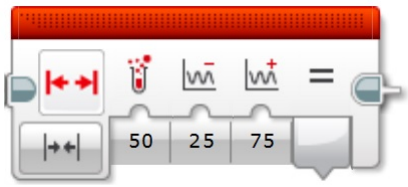


Figura 3.29 – Bloque 'Rango'

- Texto: permite combinar hasta tres variables tipo texto (cadenas de caracteres) en una única variable.

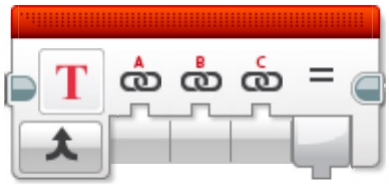


Figura 3.30 – Bloque 'Texto'

- Aleatorio: genera un valor numérico aleatorio dentro de los márgenes que le indiquemos. También puede generar un valor lógico aleatorio, pudiendo configurar el valor de probabilidad de que éste sea verdadero.

Figura 3.31 – Bloque 'Aleatorio'

3.3.1.0.5. Avanzado (pestaña azul)

- Acceso al archivo: lee/escribe en un archivo de texto que guarda en el bloque programable con el nombre que le indiquemos, lee/escribe una línea cada vez que se ejecuta el bloque. También puede configurarse para que cierre el archivo o lo elimine.

Figura 3.32 – Bloque 'Acceso al archivo'

- Registro de datos: recoge datos de los sensores que le indiquemos, puede encenderse indefinidamente, por tiempo o realizar una medida única. Puede configurarse el modo como "muestras por segundo" (0) o "segundos entre cada muestra" (1).

Figura 3.33 – Bloque 'Registro de datos'

- Mandar mensajes: Envía, recibe o compara mensajes en formato texto, numérico o lógico entre distintos bloques programables conectados mediante Bluetooth. Indicando el número identificador del bloque al que va dirigido.

Figura 3.34 – Bloque 'Mandar mensajes'

- Conexión Bluetooth: Enciende o apaga el Bluetooth, inicia y finaliza comunicación con otros bloques (indicándole el número de bloque).

Figura 3.35 – Bloque 'Conexión Bluetooth'

- Mantener activo: Sirve para mantener el robot activo si necesitamos esperar un tiempo mayor al que el bloque tiene configurado para apagarse por inactividad (este parámetro puede configurarse desde el mismo).

Figura 3.36 – Bloque 'Mantener activo'

- Valor del sensor sin procesar: indicándole el número de puerto nos devuelve el valor leído por el sensor, sin procesar, en un rango de 0 a 1023 (10 bit).

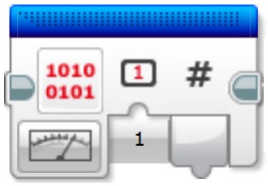


Figura 3.37 – Bloque 'Valor sin procesar'

- Motor sin regular: otorga un nivel de potencia fijo al motor, a diferencia del bloque motor 'normal', que funciona en lazo cerrado leyendo la velocidad a la que gira y ajustando la potencia en consecuencia, para que la velocidad de giro sea constante.

  Usando este bloque, al ser un sistema en lazo abierto, la velocidad puede variar en función de la inclinación o resistencia al giro del motor.

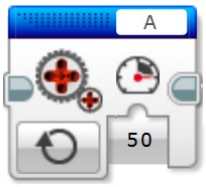


Figura 3.38 – Bloque 'Motor sin regular'

- Invertir el motor: simplemente invierte el sentido de giro.

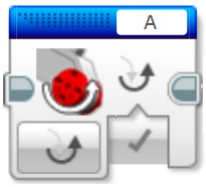


Figura 3.39 – Bloque 'Invertir el motor'

- Detener el programa: detiene la ejecución del programa.

Figura 3.40 – Bloque 'Detener el programa'

- Comentario: no realiza ninguna función de programación, sirve solamente como anotación, la diferencia con las etiquetas es que al estar integrado como bloque se mueve con el resto de bloques al arrastrar partes del código etc.

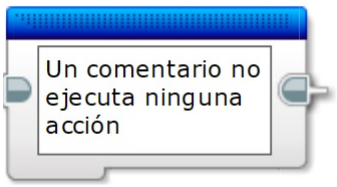


Figura 3.41 – Bloque 'Comentario'

3.3.1.0.6. Mis bloques (pestaña cian) En esta pestaña aparecerán los bloques que creemos mediante el editor de bloques. La creación de bloques o subprogramas sirve para facilitar la lectura del programa, haciéndola más sencilla, además de para aprovechar espacio en pantalla.

3.3.2. Elementos de programación: Cables de datos

Los cables de datos (traducción del inglés, data wires) sirven para intercambiar información entre los bloques. Los terminales pueden ser de entrada o salida y pueden ser de tipo numérico, texto o lógico.

En su extremo tendrán un 'conector' que dependerá del tipo de dato y de si es de entrada o salida, esto evita que realicemos interconexiones con tipos de datos incompatibles.

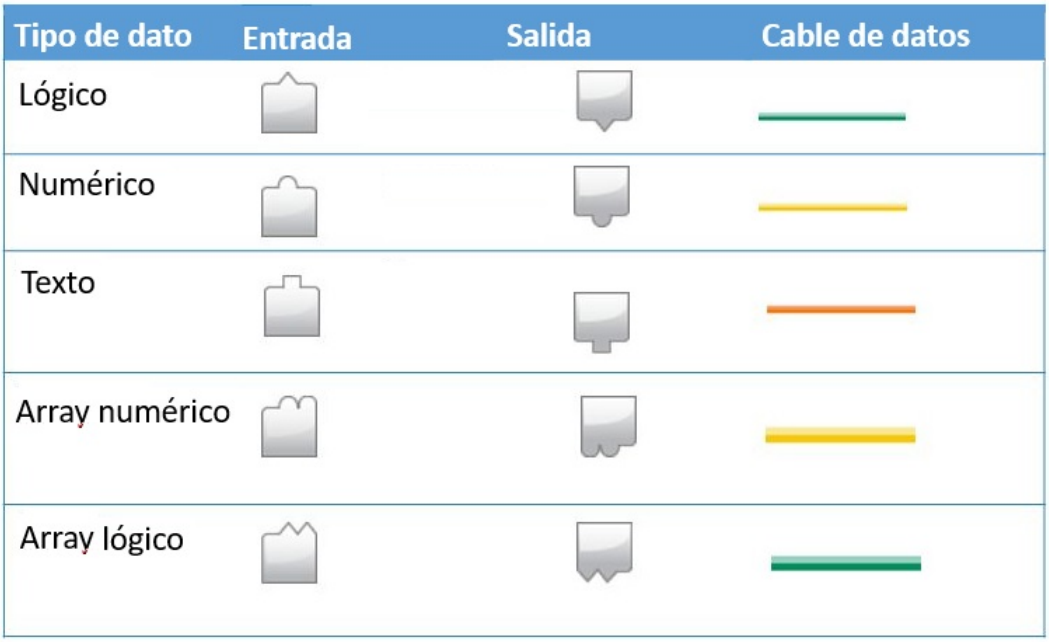


Figura 3.42 – Tipos de cables de datos

3.4. LEGO Digital Designer

LEGO Digital Designer es un software CAD desarrollado por LEGO y contiene los modelos 3D (formato .lxf) de todas las piezas fabricadas por LEGO.

La interfaz del programa es muy simple, está formada por un menú donde seleccionaremos las piezas (clasificadas por familias) y un entorno gráfico donde podremos arrastrar y ensamblar las piezas, además de visualizar el montaje desde cualquier ángulo.

En la parte superior de la ventana, disponemos de algunas funciones básicas como cambiar de color, clonar pieza, ocultar pieza etc...

Pero la opción más interesante es *'Building guide mode'*, esta opción generará las instrucciones paso a paso necesarias para el montaje del robot, a modo de tutorial.

Dado que las instrucciones son muy detalladas y la complejidad y número de piezas de algunos de los montajes es elevada, es imposible presentarlas en formato papel en el presente TFG, por motivos de espacio.

Aunque el programa nos permite exportar las instrucciones en formato html, éstas además de muy extensas, son difíciles de interpretar debido a la baja resolución de las imágenes generadas y la imposibilidad de manipular los objetos en el espacio o hacer zoom sobre ellos, por lo que la mejor opción es cargar el modelo 3D y seguir el tutorial generado por el programa.

Tanto los modelos 3D de todos los montajes, así como el software LDD (con licencia freeware) necesario para su visualización se incluyen en el CD adjunto, así como online en el enlace que se indica en el apartado 'Resultados Finales'.

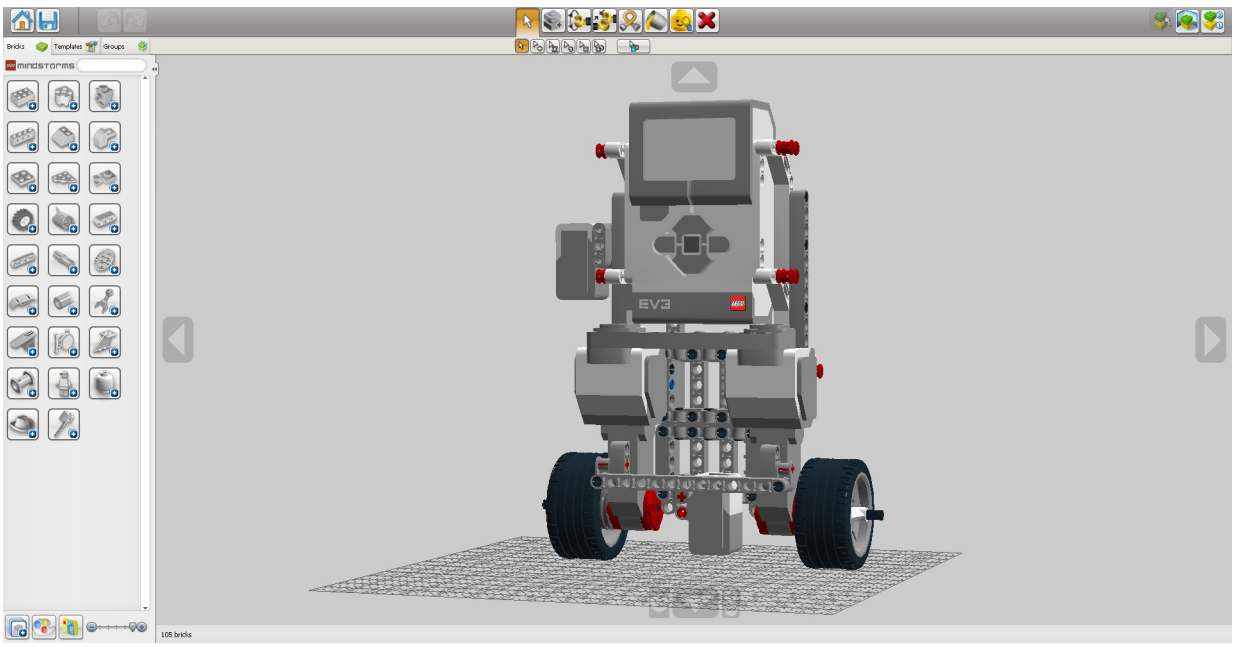


Figura 3.43 – Entorno gráfico LEGO Digital Designer

3.5. Controlador PID

Un PID es un mecanismo de control realimentado que calcula el error, esto es, la desviación entre un valor instantáneo y el valor que deseamos obtener, llamado consigna, para generar una acción correctora que ajuste el proceso. El PID debe su nombre a los tres tipos de acciones que lo conforman.

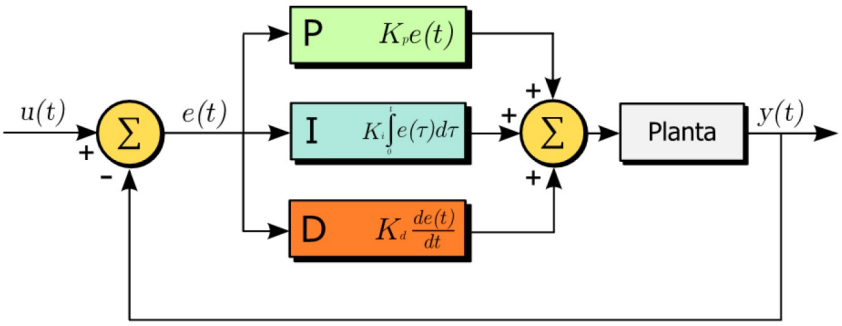


Figura 3.44 – Esquema del controlador PID

3.5.1. Acción proporcional

Decimos que la acción proporcional tiene un efecto de presente, ya que lo que hace es multiplicar el error en el instante actual por una ganancia para generar una señal de control.

El control proporcional posee error de posición, esto es debido a que a medida que la salida del sistema se aproxima a la consigna deseada, la señal de error se reduce, llegando un punto en el que la señal de control generada no puede seguir proporcionando potencia para aumentar el valor en régimen permanente de la salida, aunque sí para mantenerlo.

La acción proporcional modifica tanto el régimen transitorio como el permanente.

$$P = Kp \times e(t) \tag{3.1}$$

3.5.2. Acción integral

El control integral es una acción de pasado, acumula los valores anteriores del error, el área bajo la curva (de ahí el nombre “integral”).

Cuando la salida se encuentra por debajo del valor de consigna el error es positivo y la señal de control debe seguir proporcionando potencia al sistema para poder alcanzar dicho valor. La parte integral almacena la suma de los valores anteriores del error, por lo que la curva de la señal de control subirá superando al valor de la consigna.

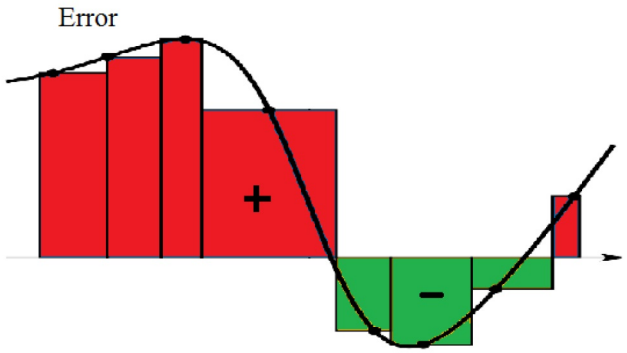


Figura 3.45 – Controlador PID: Error

En ese momento, el error pasará a ser negativo con lo cual se restará a los valores acumulados anteriormente, haciendo que la señal de control se reduzca y la curva baje de nuevo hacia el valor de consigna (3.45).

Este proceso se repite cíclicamente. La acción integral nos permite anular el error en régimen permanente.

$$I = Ki \int_0^t e(t)dt \tag{3.2}$$

3.5.3. Acción derivativa

La acción derivativa tiene un carácter de previsión (o futuro), ya que la respuesta de la componente derivativa es proporcional a la tasa de cambio del error.

Al responder a la velocidad de cambio del error produce una corrección antes de que este se vuelva demasiado grande. Esto permite hacer el sistema mucho más rápido, ya que podemos hacer más agresivas las acciones proporcional e integral. Encargándose la parte derivativa de limitar la sobreoscilación que estas introducen.

La principal desventaja de la respuesta derivativa radica en que es muy sensible al ruido en la señal de proceso, lo cual puede llegar a hacer el sistema de control inestable.

$$D = Kd\left(\frac{de(t)}{dt}\right) \tag{3.3}$$

En el apartado Anexos, se explicará en detalle el modo de obtener los parámetros del controlador PID.

4 NORMAS Y REFERENCIAS

4.1. Disposiciones legales y normas aplicadas

- Norma UNE – 1027 – 95, Plegado de planos.
- UNE-ISO 690 Información y documentación. Directrices para la redacción de referencias bibliográficas y de citas de recursos de información.

4.2. Bibliografía

[1] VALK, L., *The LEGO MINDSTORMS EV3 discovery book.*, 1ª ed. San Francisco CA, USA, No Starch Press, (2014).

[2] GARBER, G., *Learning LEGO MINDSTORMS EV3.*, 1ª ed. Birmingham, UK, Packt Publishing, (2015).

[3] HARDING, G., *Programming LEGO EV3 My Blocks.*, 1ª ed. South Bend, Indiana, USA, Apress, (2018).

[4] BASIL M. AL-HADITHI, *Sistemas Discretos de Control.*, Madrid, España, Vision Net, (2007).

4.3. Programas de cálculo

Para la realización del presente TFG se han utilizado las siguientes herramientas informáticas:

- LEGO EV3-G
- LEGO Digital Designer (Versión 4.3.11)
- Microsoft Excel
- Microsoft Word
- Autodesk AutoCAD
- TeXnicCenter

4.4. Otras referencias

[5] *Updating and resetting LEGO® MINDSTORMS® EV3 Firmware*. [Consulta 2 de agosto 2019]. Disponible en:

https://www.lego.com/en-us/service/help/products/themes-sets/mindstorms/updating-and-resetting-lego-mindstorms-ev3-firmware-408100000007884

[6] *Program Descriptions GyroBoy*. [Consulta 14 de septiembre 2019]. Disponible en:

https://le-www-live-s.legocdn.com/sc/media/files/ev3-program-descriptions/ev3-program-description-gyroboy-8f351eb6ca8c99121f27be9cb2b12ad6.pdf

[7] *Lego Mindstorms: A History of Educational Robots*. [Consulta 9 de noviembre 2019]. Disponible en: http://hackeducation.com/2015/04/10/mindstorms

[8] *Alternative Programming Languages for LEGO MINDSTORMS*. [Consulta 9 de noviembre 2019]. Disponible en:

http://www.legoengineering.com/alternative-programming-languages/

[9] *Ergonomía Laboral: ¿Cómo sentarnos correctamente frente al computador?*. [Consulta 5 de diciembre 2019]. Disponible en:

http://proikos.pe/arti-blog/439/

[10] *A PID Controller For Lego Mindstorms Robots*. [Consulta 4 de septiembre 2019]. Disponible en:

http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html

5 DEFINICIONES Y ABREVIATURAS

- LDD: Lego Digital Designer.
- SDHC: Secure Digital High Capacity, es un formato de tarjeta de memoria flash.
- GNU: Sistema operativo, formado en su totalidad por software libre.
- PID: Algoritmo de control proporcional, integral y derivativo.
- XGA: (Siglas en inglés de Extended Graphics Array) es un estándar de visualización de gráficos para ordenadores creado por IBM.

6 REQUISITOS DE DISEÑO

Para el diseño y montaje de los robots se dispondrá del siguiente material:

- Set educativo LEGO® MINDSTORMS® EV3, Ref. 45544.
- Set de expansión LEGO® MINDSTORMS® EV3, Ref. 45560.
- Motor mediano adicional LEGO® MINDSTORMS® EV3, Ref. 45503.
- Batería Recargable LEGO® DC EV3 45501.
- Cargador 10V DC LEGO® 45517.

Los diferentes montajes se ordenarán en orden creciente de dificultad, incrementando el número de actuadores/sensores y la dificultad en la programación.
Se incluirá una descripción detallada de los programas, así como las instrucciones necesarias para el montaje físico de los robots.

7 ANÁLISIS DE LAS SOLUCIONES

7.1. Robot 'Pick and Place'

7.1.1. Introducción

Construiremos un robot en forma de cinta transportadora que detecte la presencia de piezas de chapa rectangulares mediante un sensor óptico y las transporte de la cinta a un recipiente, utilizando un brazo equipado con un pequeño imán, movido por la acción coordinada de dos motores, uno que moverá el brazo hacia arriba y abajo mediante una cremallera y otro que desplazará un carrito móvil hacia delante y atrás, también mediante una cremallera.

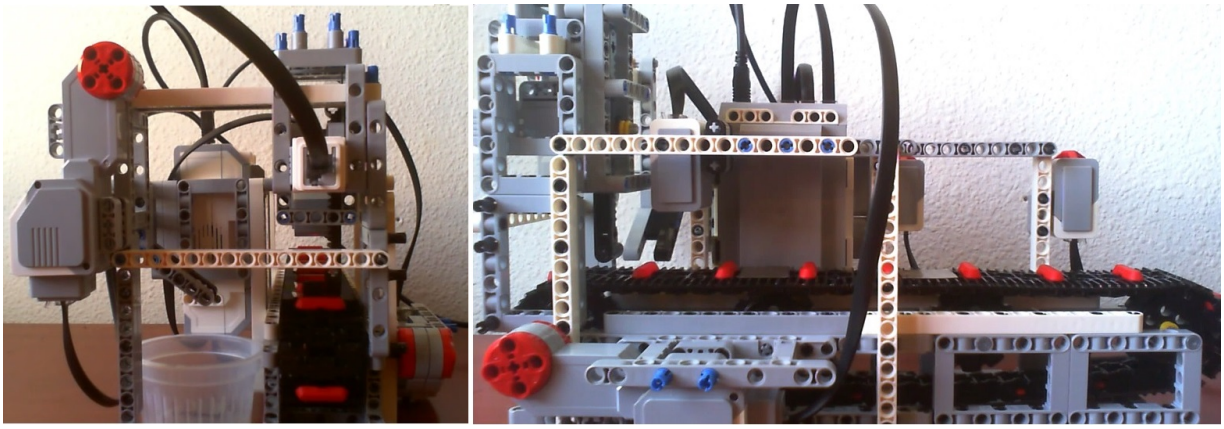


Figura 7.1 – Robot 'Pick and Place', vistas lateral y frontal.

En este montaje exploraremos las funciones:

- Uso del bloque espera.
- Uso del bucle infinito y bucle condicional.
- Uso del bloque interruptor.
- Uso de los bloques motor mediano y motor grande.

7.1.2. Sensores y actuadores

Utilizaremos como sensores:

- Dos sensores táctiles a modo de pulsadores (Puertos 1 y 2).
- Un sensor óptico para comparar la luz reflejada (Puerto 3).

Utilizaremos como actuadores:

- 1 motor grande para mover la cinta (puerto C).

- 1 motor grande para el carrito (puerto B).
- 1 motor mediano para subir y bajar el brazo (puerto A).

7.1.3. Descripción del programa

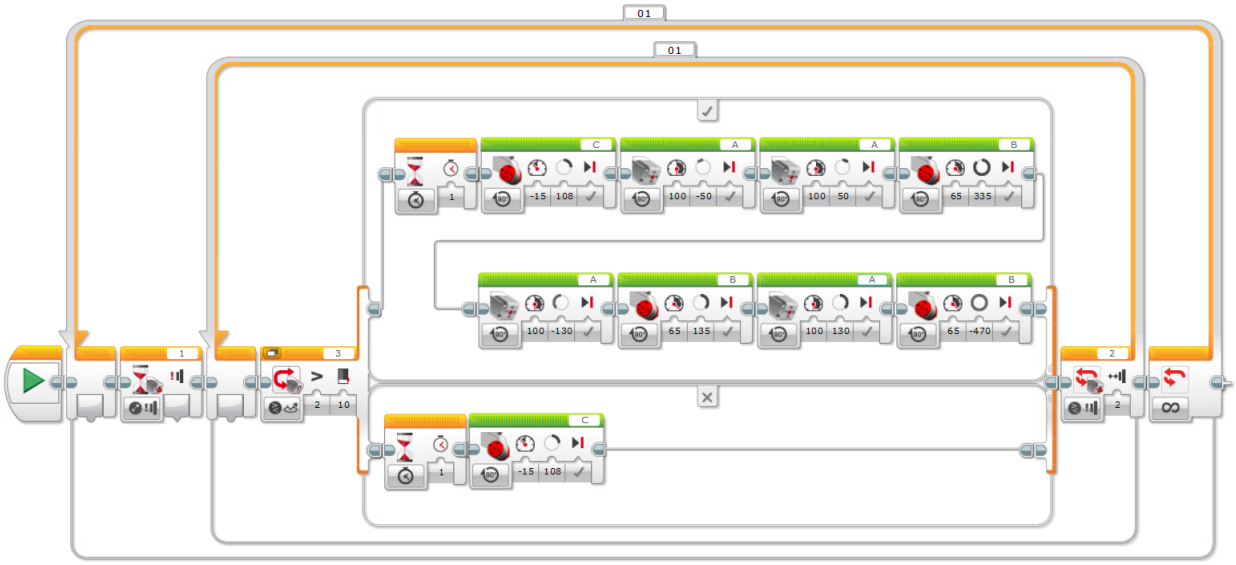


Figura 7.2 – Estructura del programa 'Pick and Place'.

Comienza, al igual que todos los programas, por el bloque iniciar. Inmediatamente después colocamos un bucle que se ejecuta un número infinito de veces.

Dentro de este bucle infinito colocamos un bloque de espera, este bloque espera a que se active el sensor táctil conectado al puerto 1 para seguir con el flujo del programa.

Tras el bloque de espera conectamos otro bucle que se ejecuta continuamente hasta que se presiona el sensor táctil conectado al puerto 2. Dentro de este bucle conectamos un bloque tipo interruptor, este bloque evalúa si el nivel de luz reflejada captada por el sensor es mayor a 10 (en una escala 1 – 100), según se cumpla esta condición existen dos posibilidades:

- Si es verdadero:

    1. Espera 1 segundo.
    2. Gira el motor de la cinta 108º hacia delante con una potencia de 15.
    3. Gira el motor de la cremallera hacia abajo 50º con una potencia de 100.
    4. Gira el motor de la cremallera hacia arriba 50º con una potencia de 100.
    5. Gira el motor que mueve el carrito 335º hacia atrás con una potencia de 65.
    6. Gira el motor de la cremallera hacia abajo 130º con una potencia de 100.
    7. Gira el motor que mueve el carrito otros 135º hacia atrás con una potencia de 65.

8. Gira el motor de la cremallera hacia arriba 130º con una potencia de 100, haciendo que la pieza tropiece y se separe del imán, cayendo en un recipiente.
9. Gira el motor que mueve el carrito 470º hacia delante con una potencia de 65, haciendo así que regrese a la posición inicial.

- Si es falso:

1. Espera 1 segundo.
2. Gira el motor de la cinta 108º hacia delante con una potencia de 15.

7.2. Robot esquiva-obstáculos

7.2.1. Introducción

El objetivo de este montaje es la realización de un robot que evite obstáculos.

El robot se desplazará hacia delante hasta encontrar un obstáculo, se detendrá, hará girar el sensor ultrasónico a izquierda y derecha mediante un motor, tomando sendas medidas, guardándolas en variables y comparándolas, en función de lo cual, girará 90º hacia el lado donde la distancia sea mayor.

Este proceso se repetirá indefinidamente hasta que el lector óptico detecte una franja de color negro en el suelo, a modo de línea de detención.

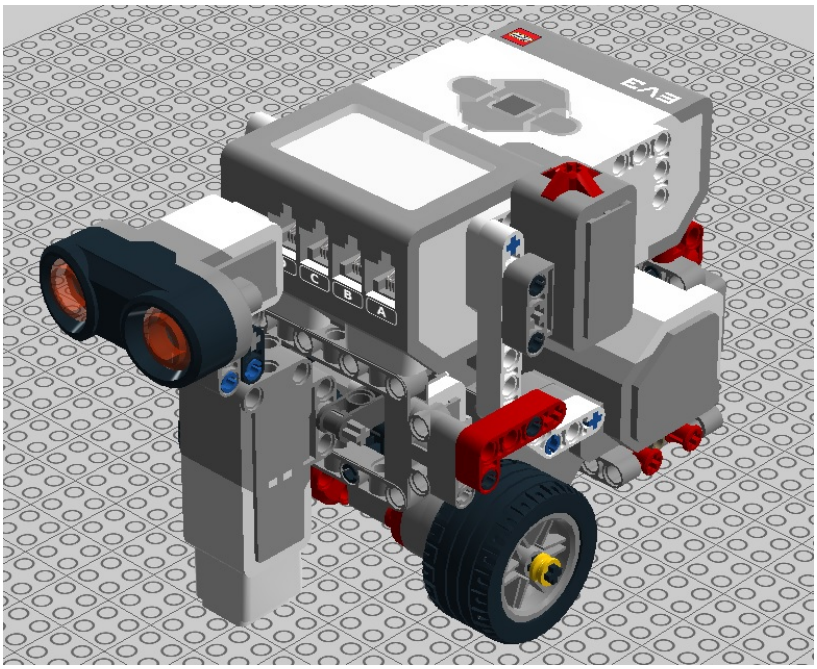


Figura 7.3 – Modelo 3D del robot esquiva obstáculos.

Implementaremos el uso de los siguientes elementos/funciones:

- Variables (numéricas).
- Bloque comparador.
- Cables de datos.
- Interrupción de bucle
- Programas simultáneos.

7.2.2. Sensores y actuadores

Utilizaremos como sensores:

- Un sensor ultrasónico que mida la distancia al obstáculo (Puerto 1).
- Un sensor óptico que detecte el color negro y detenga el robot (Puerto 2).
- Un sensor giroscópico para controlar la amplitud de los giros (Puerto 3).
- Un sensor táctil a modo de pulsador (Puerto 4).

Utilizaremos como actuadores:

- Dos motores grandes para mover el robot (Puertos B y D).
- Un motor mediano para hacer girar el sensor ultrasónico (Puerto A).

7.2.3. Variables

Nombre	Tipo	Descripción
izq	Num	Almacena el valor de la distancia a la izquierda del robot
drcha	Num	Almacena el valor de la distancia a la derecha del robot

Tabla 7.1 – Robot esquiva obstáculos: Variables

7.2.4. Descripción del programa

El programa comienza con el bloque iniciar, tras él, un bucle infinito (01) permite que se ejecute indefinidamente, el programa permanece parado hasta que se pulsa el sensor táctil (Puerto 4) como orden de inicio, cuando esto sucede, entra dentro de otro bucle infinito (02) que contiene la secuencia de pasos a ejecutar.

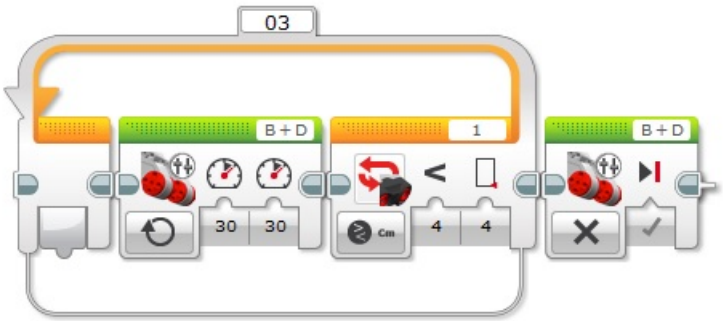


Figura 7.4 – Robot esquiva obstáculos, avance y paro.

Una vez dentro del bucle 02, encontramos un bucle condicional (03), mueve el tanque hacia delante hasta que el sensor ultrasónico detecta que se encuentra a una distancia menor de 4cm, cuando esto sucede, sale del bucle y detiene los motores (7.4).

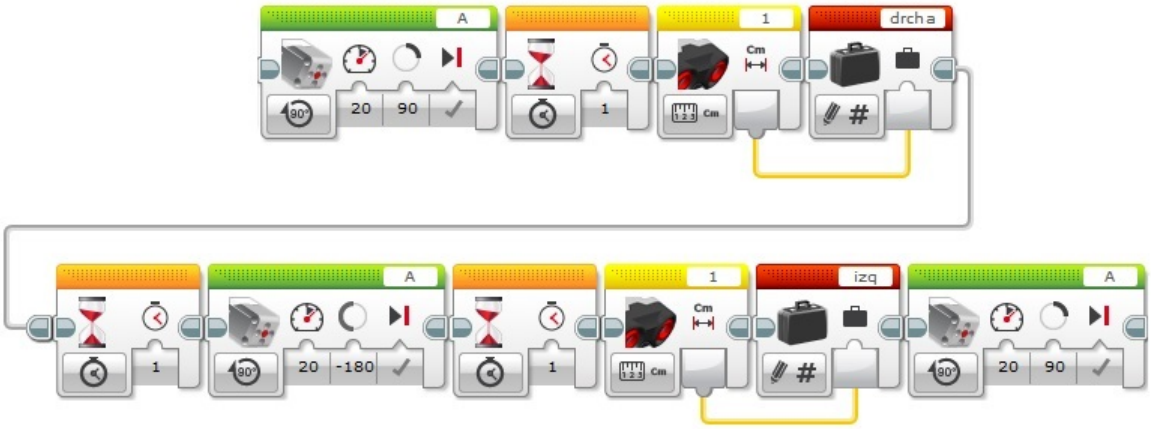


Figura 7.5 – Robot esquiva obstáculos, medición de la distancia.

Acto seguido el motor mediano A hace girar el sensor ultrasónico hacia la derecha 90º y guarda la lectura del sensor en la variable numérica «drcha».

Gira 180º hacia la izquierda y guarda la lectura en la variable «izq». Los bloques de espera de 1s sirven para que la lectura se tome con el motor que mueve el sensor parado y sea más fiable.

Un giro de 90º hacia la derecha devuelve el sensor a la posición inicial (7.5).

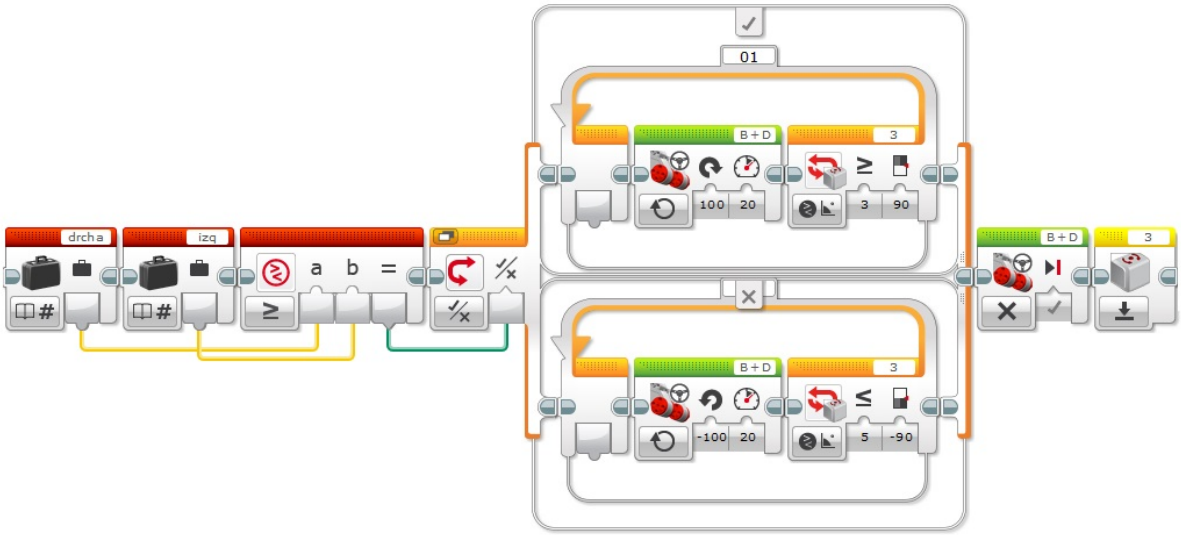


Figura 7.6 – Robot esquiva obstáculos, comparación.

A continuación, lee las variables «drcha» e «izq» y las compara mediante el bloque correspondiente, la salida de tipo lógico se conecta a un bloque tipo interruptor. Si la variable «drcha» es mayor que «izq», el robot girará 90º hacia la derecha, de lo contrario, 90º hacia la izquierda.

En ambos casos utilizamos un bucle que activa el bloque “mover la dirección” en el sentido que corresponda, hasta que el giroscopio lea los grados correspondientes (90 o -90). Tras esto paramos los motores y reiniciamos la lectura del giroscopio (7.6).

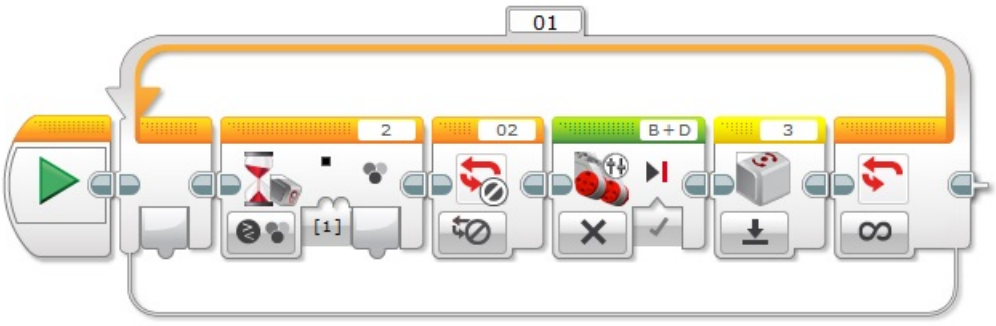


Figura 7.7 – Robot esquiva obstáculos, detención.

De manera paralela, en todo momento, el programa ejecuta un bucle infinito que evalúa si el sensor óptico detecta el color negro ([1]), si es así, sale del bucle 02, apaga los motores y reinicia la lectura del giroscopio (7.7).

7.3. Robot seguidor de línea

7.3.1. Introducción

Realizaremos el montaje de un robot móvil, que se desplace siguiendo una línea, tomando lecturas de luz reflejada por el sensor óptico e implementando un algoritmo PID mediante bloques matemáticos. El robot se detendrá cuando detecte un obstáculo a una distancia inferior a 10 cm y reanudara la marcha cuando activemos un pulsador.

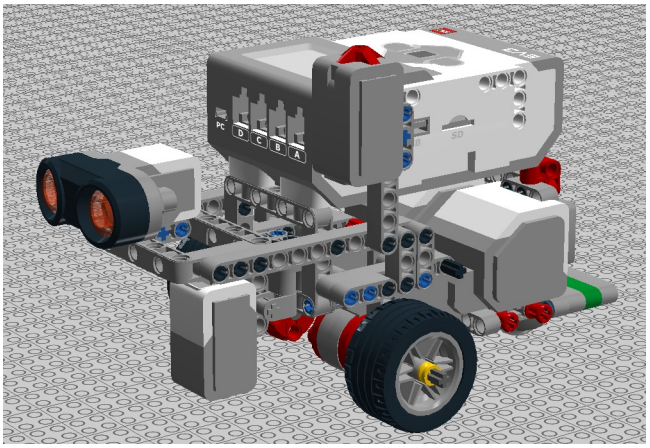


Figura 7.8 – Modelo 3D del robot seguidor de línea.

Con este montaje se pretende demostrar el uso de los bloques matemáticos.

La proporción de luz reflejada por el área iluminada por el sensor óptico nos da una idea de la posición del robot. Realizamos una medida centrando el sensor en el borde exterior de la banda negra y la superficie blanca, obteniendo un valor de 35, éste será nuestro valor de SetPoint o consigna.

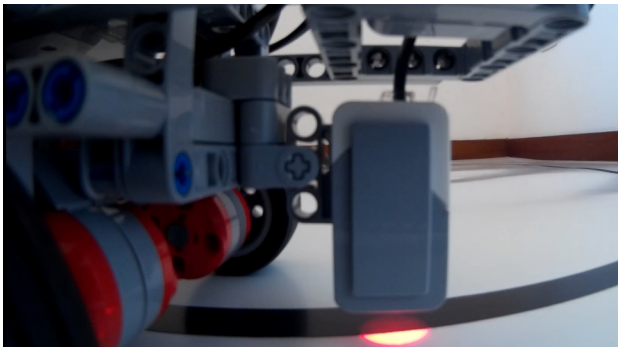


Figura 7.9 – Detalle del sensor óptico.

Si el área iluminada aumenta, el valor de la lectura será mayor de 35 y significaría que el robot va hacia la derecha. El caso contrario indicaría que va hacia la izquierda (7.9).

7.3.2. Sensores y actuadores

Utilizaremos como sensores:

- Un sensor táctil para dar la orden de arranque (Puerto 1).
- Un sensor ultrasónico para medir distancia (Puerto 2).
- Un sensor óptico para medir la intensidad de luz reflejada (Puerto 3).

Utilizaremos como actuadores:

- Dos motores grandes para mover el robot (Puertos A y D).

7.3.3. Variables

Nombre	Tipo	Descripción
SetPoint	Num	Valor de consigna.
Error	Num	Diferencia entre el valor leído y el valor de consigna.
Error ant	Num	Valor del error en la iteración anterior.
Integral	Num	Valor del error acumulado.
Derivada	Num	Diferencia entre «Error» y «Error ant».
Kp	Num	Término proporcional del PID.
Ki	Num	Término integral del PID.
Kd	Num	Término derivativo del PID.

Tabla 7.2 – Robot seguidor de línea: Variables

7.3.4. Descripción del programa

El programa se ejecuta continuamente dentro de un bucle infinito, dentro de este bucle colocamos un bloque de espera condicionado a la pulsación del sensor táctil.

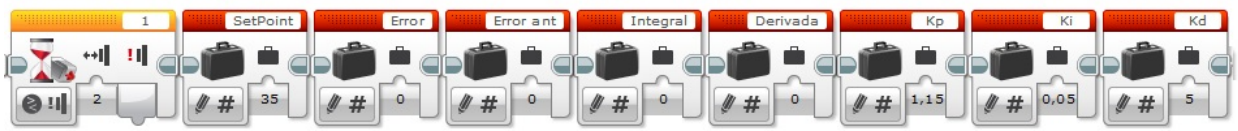


Figura 7.10 – Robot seguidor de línea, inicialización de variables.

Una vez pulsado, damos valor 0 a las variables «Error», «Error ant», «Integral» y «Derivada», y cargamos los valores en las variables «SetPoint», «Kp», «Kd» y «Ki» (7.10).

«Kp», «Ki», «Kd»: Otorgan pesos a las componentes proporcional, integral y derivativa del algoritmo PID.

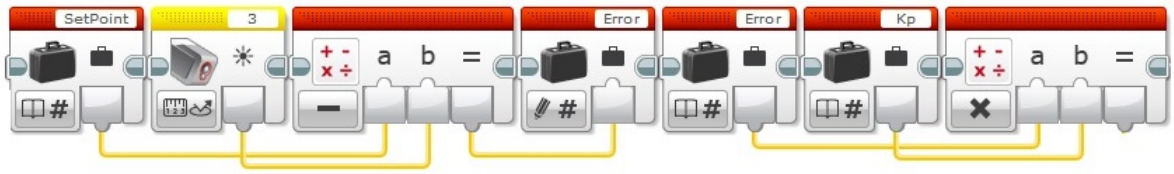


Figura 7.11 – Robot seguidor de línea, término proporcional.

Tras asignar estos valores, el programa entra en un bucle que se detiene cuando el sensor ultrasónico detecta un objeto a una distancia inferior a 10 cm.

Una vez en el bucle, comenzamos restando el valor de «SetPoint» a la lectura del sensor óptico, volcando el resultado en la variable «Error», multiplicamos el error por el coeficiente «Kp», obteniendo de este modo el término proporcional del algoritmo PID (7.11).

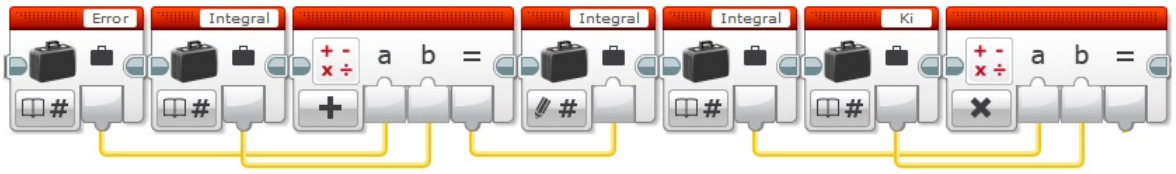


Figura 7.12 – Robot seguidor de línea, término integral.

A continuación, a la variable «Error» le sumamos el valor de «Integral» y lo volcamos de nuevo sobre la variable «Integral», actualizándola, multiplicando este valor por el coeficiente «Ki», obtendremos el valor del término integral del algoritmo (7.12).

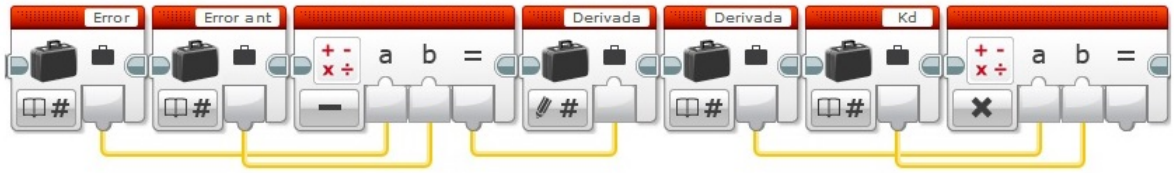


Figura 7.13 – Robot seguidor de línea, término derivativo.

De forma análoga, en el siguiente segmento de bloques, restamos el valor de la variable «Error ant» al de la variable «Error», volcando el resultado en la variable «Derivada», multiplicando el valor de «Derivada» por el coeficiente Kd, tendremos el valor del término derivativo (7.13).

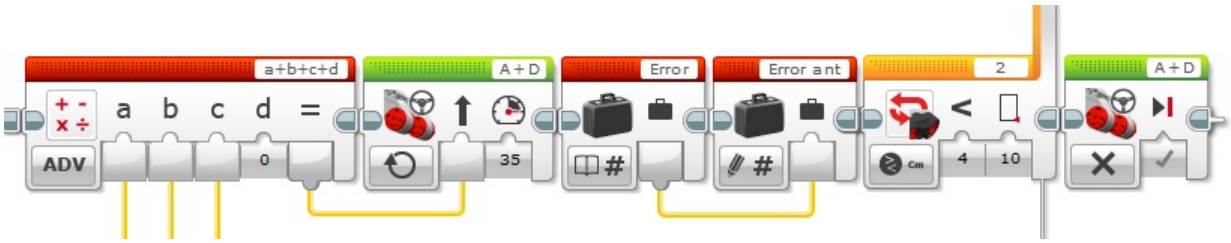


Figura 7.14 – Robot seguidor de línea, suma de términos.

Sumamos los tres términos, proporcional, integral y derivativo, el resultado de esta operación será el parámetro que enviemos al bloque encargado de mover la dirección (elegimos un nivel de potencia de 35).

Por último, volcamos el valor de «Error» en «Error ant», actualizándolo. En caso de que se cumpla la condición de salida del bucle (objeto a distancia menor de 10 cm), detendríamos los motores y volveríamos de nuevo al inicio. (7.14).

7.4. Robot seguidor de línea - Recogida de datos

7.4.1. Introducción

Sobre la base del robot anterior, con el mismo funcionamiento y añadiendo un sensor giroscópico, exploraremos las funciones de recogida y exportación de datos, así como la creación de bloques a modo de subprogramas.

La función de recogida de datos nos permite almacenar e indexar la información recogida por los sensores frente al tiempo. Analizando estos datos posteriormente, bien en el entorno EV3 o en un programa externo, podremos analizar y optimizar el desempeño del robot (en la sección Anexos se indica cómo optimizar los parámetros del seguidor PID utilizando este recurso).

7.4.2. Mi bloque

El uso de bloques personalizados, nos permite crear subprogramas contenidos en un solo bloque, asignándole variables de entrada y salida si es necesario, esto simplifica enormemente la lectura a simple vista del diagrama de bloques, reduciendo su tamaño.

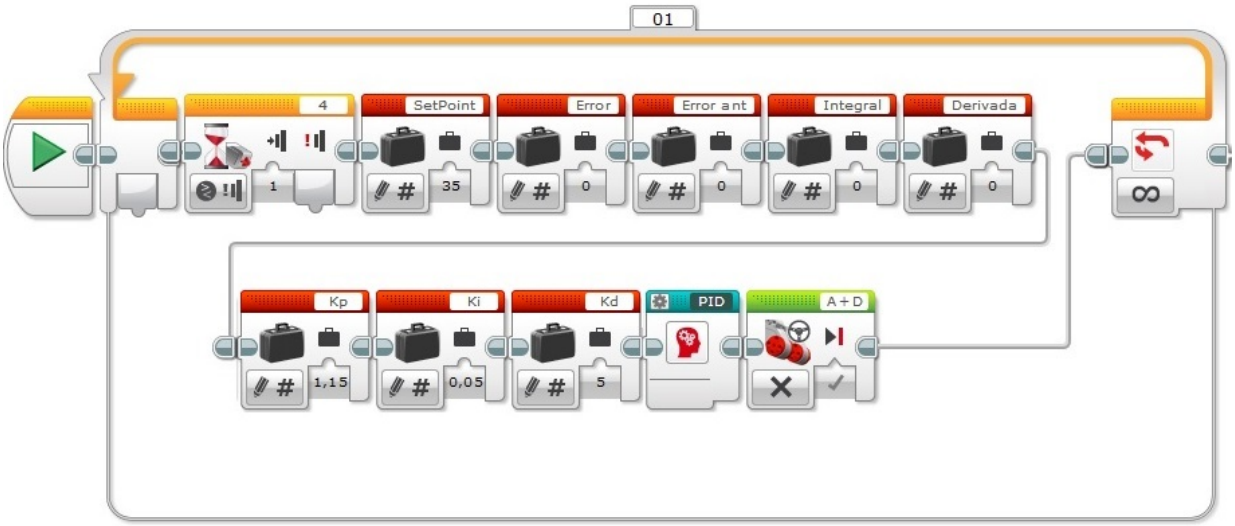


Figura 7.15 – Robot seguidor de línea, utilizando ”Mi Bloque”

En el ejemplo 7.15 hemos tomado el programa anterior y hemos creado un bloque llamado PID que contiene el bucle que ejecuta el algoritmo. Para ello, basta con seleccionar los bloques y dirigirnos al menú ’Herramientas’/’Constructor de Mi Bloque’.

7.4.3. Recogida de datos

Figura 7.16 – Robot seguidor de línea, recogida de datos

Usando el bloque avanzado ’Registro de Datos’ (7.16), podemos tomar lecturas de los sensores y encoders, eligiendo entre dos modos:

- Muestras por segundo (opción 0).
- Segundos entre cada muestra (opción 1).

En nuestro caso escogeremos la primera opción y tomaremos 10 muestras por segundo.

Los datos se guardarán en un archivo formato .rdf, con el nombre que especifiquemos en la esquina superior izquierda del bloque, en nuestro caso “Datos”.

7.4.4. Visualización de los datos

Para visualizar los datos recogidos, en el menú 'Archivo', seleccionaremos 'Proyecto nuevo'/'Experimento'.

A continuación en el menú 'Herramientas', seleccionaremos 'Administrador de archivos de registro de datos' y buscaremos el archivo de datos correspondiente.

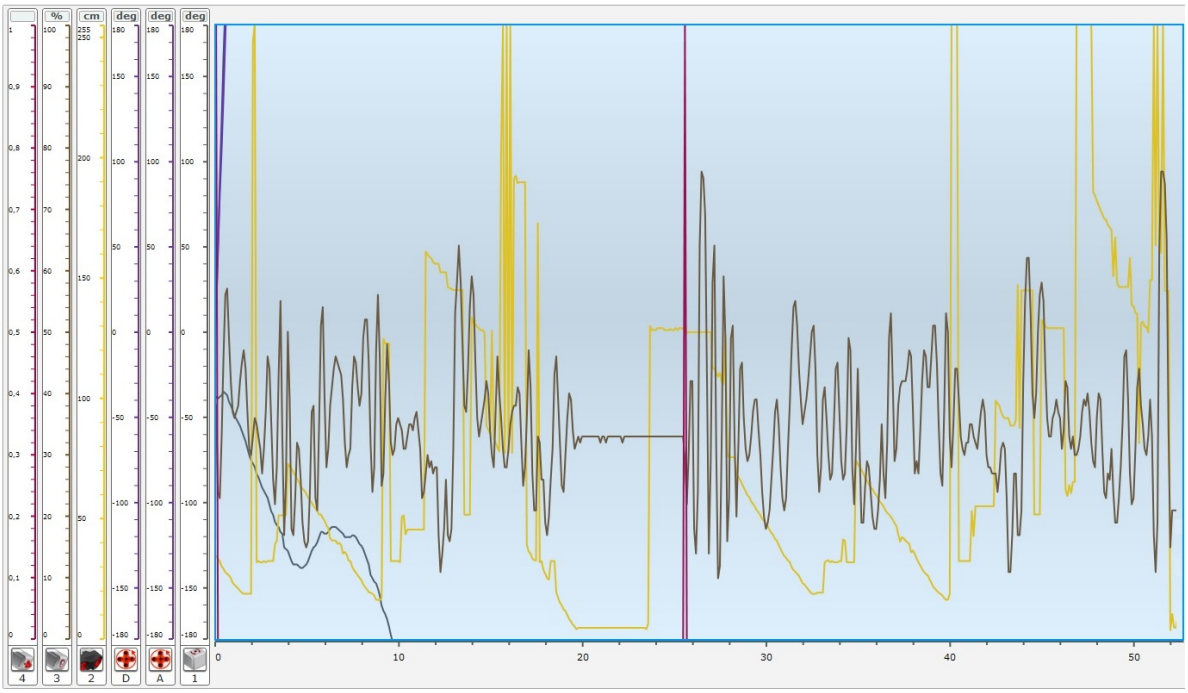


Figura 7.17 – Robot seguidor de línea, visualización de los datos.

Al cargarlo, generará un gráfico con los valores de los sensores/encoders frente al tiempo, podremos ajustar la escala y seleccionar los datos que queremos visualizar(7.17).

También nos da la opción de realizar operaciones matemáticas básicas con los conjuntos de datos, como búsqueda de máximos y mínimos, multiplicación, suma, etc. . .

Esta herramienta es útil en el caso de programas sencillos. Si el número de sensores o el volumen de datos que estamos visualizando es elevado, o si queremos realizar operaciones más complejas sobre el conjunto de datos, la mejor opción es exportar los datos a una hoja de cálculo.

Para ello, en el menú 'Herramientas', seleccionamos 'Exportar conjunto de datos', genera un archivo .csv que puede ser abierto con una aplicación de hojas de cálculo.

	A	B	C	D	E	F	G
1	Time	Girosensor_p	Rotación_pA	Rotación_pD	Ultrasónico_p	Color_p3_01	Táctil_p4_01
2	0	-38	0	-7	34,9	39	1
3	0,1	-38	15	10	34,7	38	1
4	0,2	-39	49	46	33,4	24	0
5	0,3	-38	88	80	32,6	23	0
6	0,4	-37	124	107	30,4	33	0
7	0,5	-35	161	138	29,4	46	0
8	0,6	-36	191	172	28	56	0
9	0,7	-37	213	206	27,3	57	0
10	0,8	-41	231	243	26,7	49	0
11	0,9	-43	247	275	25,5	42	0
12	1	-46	268	312	23,6	37	0

Figura 7.18 – Robot seguidor de línea, datos en Excel.

Nos mostrará la variable tiempo en la primera columna, seguido de las medidas de cada sensor/encoder con su correspondiente número de puerto en las siguientes columnas(7.18).

Utilizando una hoja de cálculo nos será más sencillo visualizar y procesar los datos y gráficos que en el entorno del programa EV3-G.

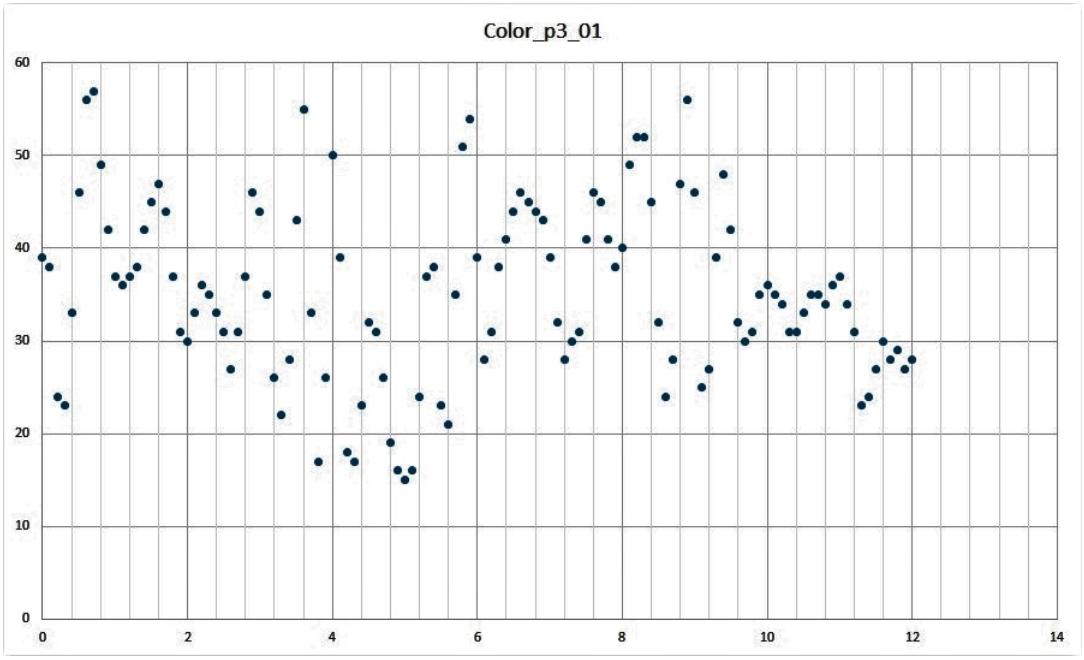


Figura 7.19 – Robot seguidor de línea, gráfico: intensidad luminosa frente a tiempo.

7.5. Robot 'Telesketch'

7.5.1. Introducción

Realizaremos un montaje que nos permita trazar líneas rectas en la pantalla del bloque programable, utilizando la lectura de los encoders de los motores. Mediante los botones del bloque, seleccionaremos el grosor de la línea y el modo (Escribir/Borrar).

Figura 7.20 – Robot «Telesketch».

En este montaje exploraremos el uso de variables lógicas (o bool), la lectura de encoders y el uso del bloque 'Pantalla'.

7.5.2. Sensores y actuadores

Utilizaremos los encoders de los motores A y D para mover el cursor en dirección horizontal y vertical respectivamente.

Utilizaremos también un sensor giroscópico (puerto 1) y un sensor táctil (puerto 2).

7.5.3. Variables

Nombre	Tipo	Descripción
X	Num	Almacena el valor de la coordenada X.
Y	Num	Almacena el valor de la coordenada Y.
tamano	Num	Modifica el grosor de línea.
escrib	Bool	Habilita el modo escritura.
borra	Bool	Habilita el modo borrado.

Tabla 7.3 – Robot 'Telesketch': Variables

7.5.4. Descripción del programa

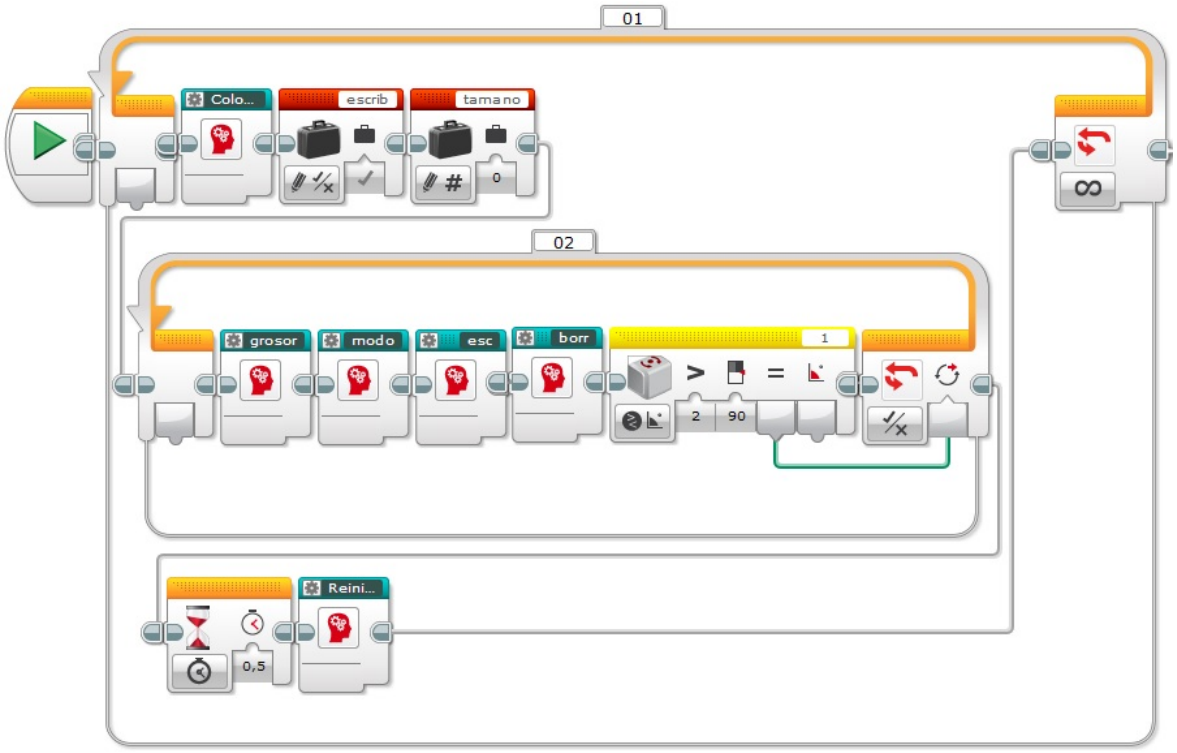


Figura 7.21 – Estructura del programa «Telesketch».

El programa se ejecuta dentro de un bucle infinito (7.21), en primer lugar, ejecuta el bloque 'Colocar', moveremos las ruedas conectadas a los motores, para fijar la posición inicial del cursor. Una vez fijada, pulsamos el sensor táctil y podremos empezar a trazar la línea.

El modo por defecto es el de escritura y el tamaño de línea por defecto 0.

Los botones arriba y abajo del bloque programable nos permiten aumentar y disminuir el grosor de la línea. Los botones izquierda y derecha del bloque programable sirven como selección de modo, Escribir/Borrar.

Moviendo el robot un ángulo mayor de 90º, se borrará la pantalla y se reiniciarán lecturas de encoders y variables, permitiendo empezar de nuevo.

7.5.4.0.1. Bloque 'Coordenada' : Este bloque se utilizará en los bloques 'Colocar', 'esc' (escribir) y 'borr' (borrar).

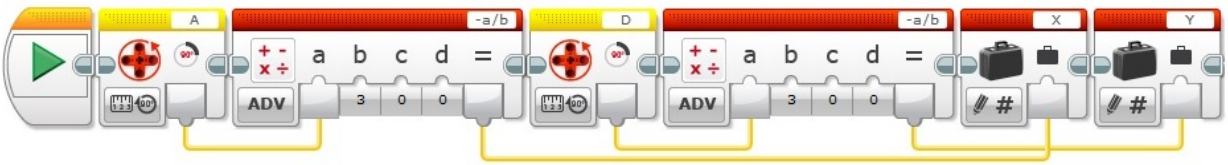


Figura 7.22 – Bloque 'Coordenada': lectura de encoders.

Toma la lectura de los encoders A y D, divide esta lectura entre 3 (este valor sirve para ajustar la sensibilidad) y almacena el resultado en las variables «X» e «Y», respectivamente (7.22).

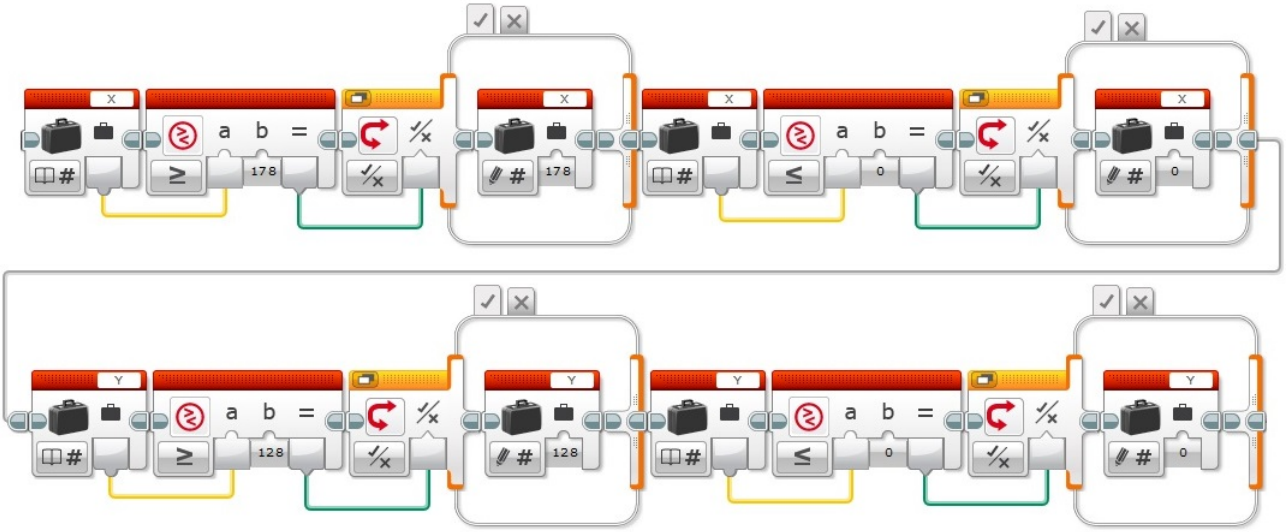


Figura 7.23 – Bloque 'Coordenada': limitación del rango de X e Y.

A continuación limita el rango de X e Y para que se corresponda con la dimensión en píxeles de la pantalla (7.23).:

- Si X es mayor de 178, toma el valor 178. Si es menor de 0 toma el valor 0.
- Si Y es mayor de 128, toma el valor 128. Si es menor de 0 toma el valor 0.

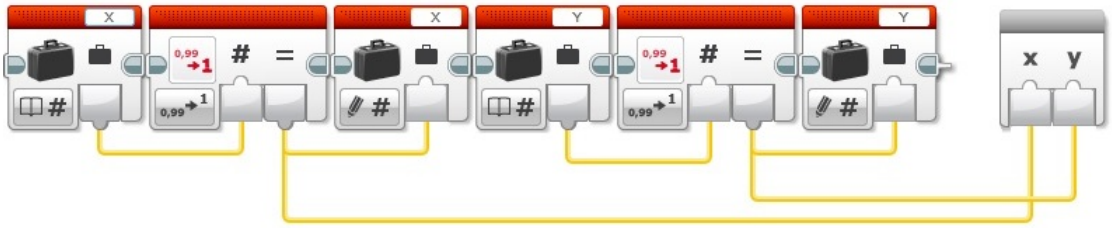


Figura 7.24 – Bloque 'Coordenada': redondeo de valores X e Y.

Por último, redondea los valores de X e Y al valor entero más cercano (7.24).

7.5.4.0.2. Bloque 'Colocar' : Sirve para colocar el cursor en la posición de la pantalla en la que queremos empezar a dibujar.

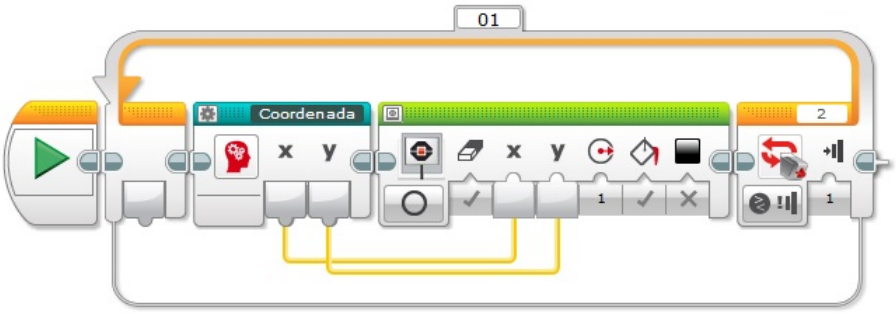


Figura 7.25 – Bloque 'Colocar'.

Se trata de un bucle que se ejecuta hasta que se pulsa el sensor táctil (Puerto 2), toma la salida del bloque 'Coordenada' e imprime un punto en la pantalla en las coordenadas indicadas (7.25).

7.5.4.0.3. Bloque 'Grosor' : Modifica el grosor de línea.

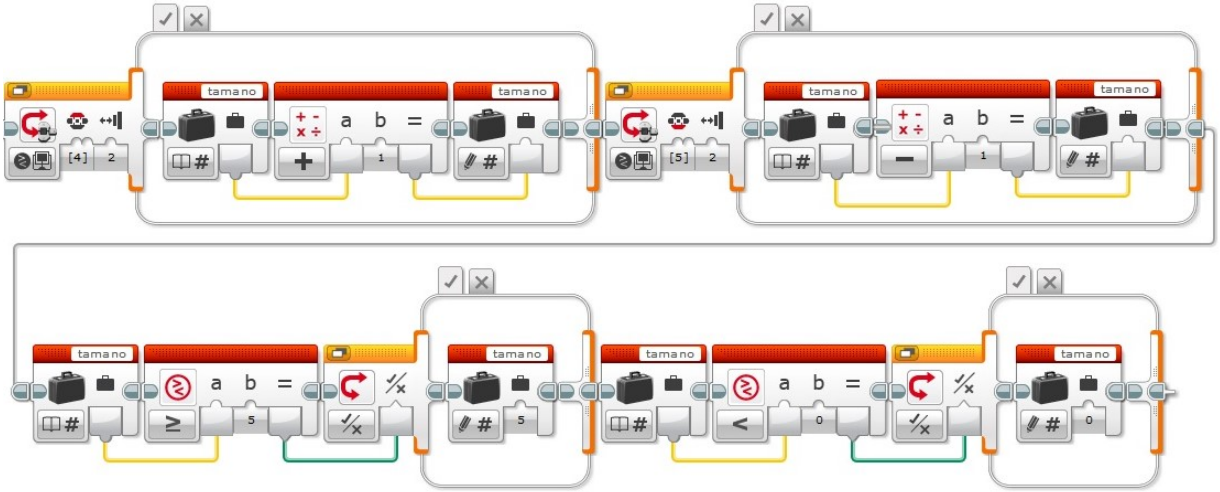


Figura 7.26 – Bloque 'Grosor'.

Al pulsar el botón superior de la botonera (4), la variable «tamano» se incrementa en una unidad, de manera análoga, al pulsar el botón inferior (5), decrementa una unidad. A continuación acota la variable «tamano» entre un valor máximo de 5 y mínimo de 0 (7.26).

7.5.4.0.4. Bloque 'Modo' : Determina el modo (borrar/escribir).

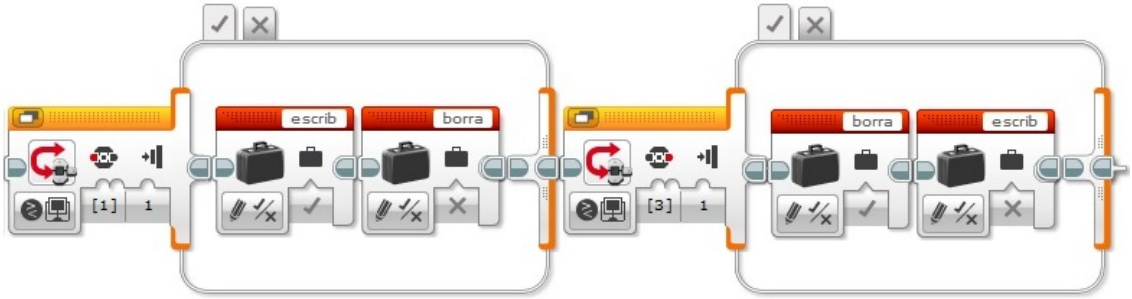


Figura 7.27 – Bloque 'Modo'.

Pulsando el botón izquierdo de la botonera (1), la variable «escrib» toma valor verdadero y la variable «borra», valor falso. La acción contraria sucede al pulsar el botón derecho (3) (7.27).

7.5.4.0.5. Bloque 'Esc' : Realiza la función de escritura.

Figura 7.28 – Bloque 'Esc'.

Si la variable lógica «escrib» tiene valor verdadero, imprime en pantalla un punto de color negro del grosor indicado por la variable «tamano». Como la opción borrar pantalla está configurada como falso, la sucesión de puntos genera una línea (7.28).

7.5.4.0.6. Bloque 'Borr' : Realiza la función de borrado. Este bloque funciona de manera análoga al anterior, evaluando en este caso la variable «borra», en esta ocasión el color seleccionado es blanco, por lo que al sobrescribir crea el efecto de borrado.

7.5.4.0.7. Bloque 'Reiniciar' : Este bloque reinicia las lecturas de los encoders A y D y del giroscopio, da valor 0 a las variables «X», «Y» y «tamano» y borra la pantalla.

7.6. Robot 'Trazador'

7.6.1. Introducción

Este montaje, reutiliza parte del código anterior, utilizando la lectura de los encoders, permite realizar un trazado mediante líneas rectas en la pantalla del bloque programable, almacenando los valores de las coordenadas X e Y en sendos archivos de texto, para que posteriormente el robot lea estos archivos y se mueva replicando dicho trazado.

El principio en que se basa el algoritmo es comparar dos lecturas consecutivas de X e Y para detectar cuándo se produce un cambio de dirección en el trazado y en qué dirección debe girar.

En este montaje exploraremos los bloques de lectura y escritura de archivos.

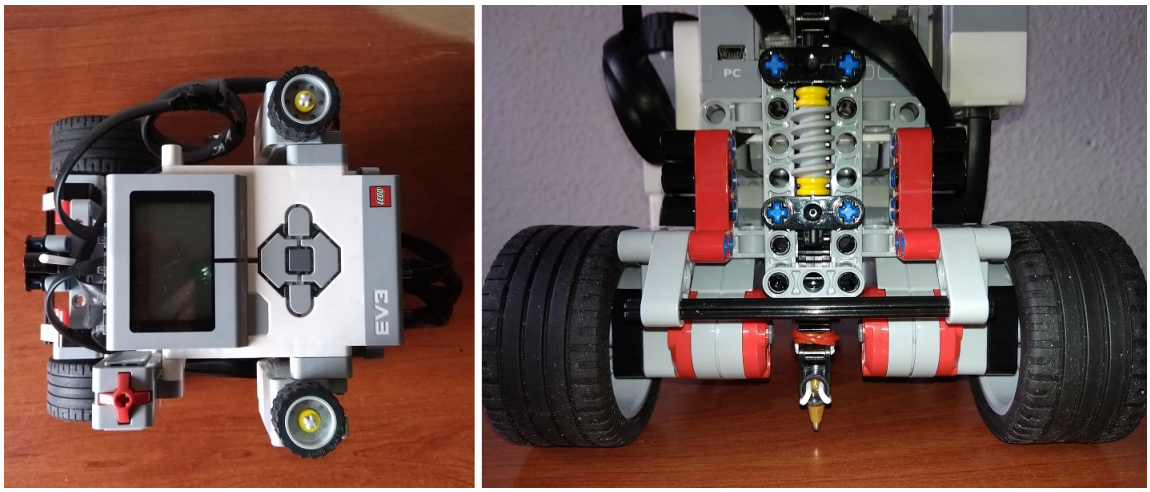


Figura 7.29 – Robot trazador: vista superior y frontal.

7.6.2. Sensores y actuadores

Utilizaremos como sensores los encoders de los motores medianos C y D, para trazar las líneas en pantalla, de manera análoga al ejemplo anterior. Dispondremos también de un sensor táctil (puerto 2).

Utilizaremos como actuadores dos motores grandes conectados a los puertos A y B para mover el robot.

7.6.3. Variables

Nombre	Tipo	Descripción
X	Num	Almacena el valor de la coordenada X.
Y	Num	Almacena el valor de la coordenada Y.
X-1	Num	Almacena el valor anterior de la coordenada X.
Y-1	Num	Almacena el valor anterior de la coordenada Y.

Tabla 7.4 – Robot trazador: Variables

7.6.4. Descripción del programa

El programa se ejecuta dentro de un bucle infinito, dentro de este bucle, se bifurca en dos ramas.

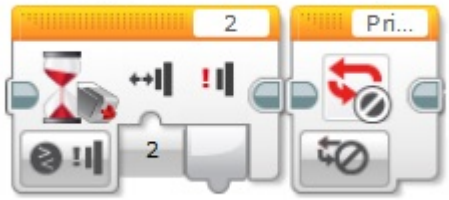


Figura 7.30 – Robot trazador: interrupción de bucle

Una rama que contiene el programa en sí, en un bucle llamado 'Principal', y otra que activa una interrupción de dicho bucle si se activa el pulsador (7.30).

El bucle principal comienza por el bloque 'Inicio', que reinicia lecturas de sensores y variables, a continuación entra en el bucle 'Comienzo', que al igual que en el ejemplo anterior, sirve para colocar el cursor en la pantalla. Este bucle se ejecuta hasta que pulsamos el botón central del bloque programable.

A continuación, entra en el bucle 'lectura', este bucle nos permite realizar el trazado en la pantalla de forma análoga al ejemplo anterior, con la diferencia de que almacena las coordenadas en dos archivos de texto llamados «ejex» y «ejey», para su posterior lectura.

Realiza dos lecturas sobre los archivos «ejex» y «ejey», almacenando de este modo dos valores consecutivos de la coordenada X e Y.

El bucle 'movimiento', haciendo uso del bloque comparador y de su salida lógica, compara valores consecutivos de X e Y, para determinar si el robot debe avanzar o girar y en qué sentido, los giros en ángulo recto se controlan mediante el sensor giroscópico.

7.6.4.0.1. Bucle 'Comienzo' : Toma los valores de X e Y del bloque Coordenada (este bloque es idéntico al utilizado en el ejemplo anterior) y los envía al bloque Pantalla, para imprimir

un cursor.

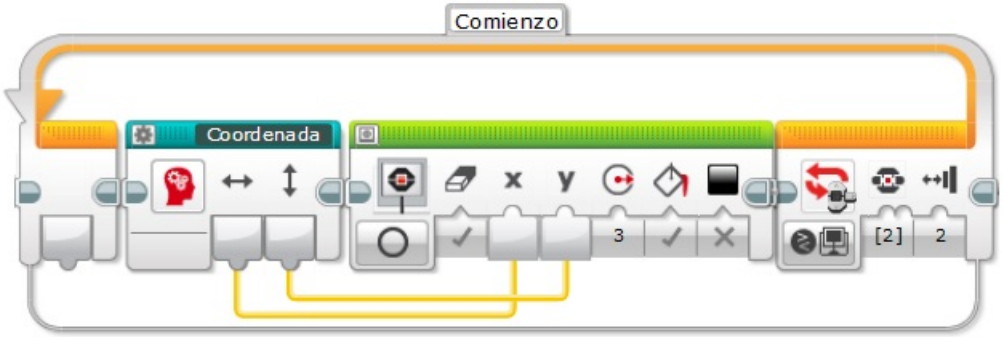


Figura 7.31 – Robot trazador: Bucle 'Comienzo'

El bucle se interrumpe al pulsar el botón central del brick (7.31).

7.6.4.0.2. Bucle 'lectura' : Una vez colocado el cursor en la posición deseada y pulsado el sensor, este bucle almacenará la posición del mismo y trazará la línea.

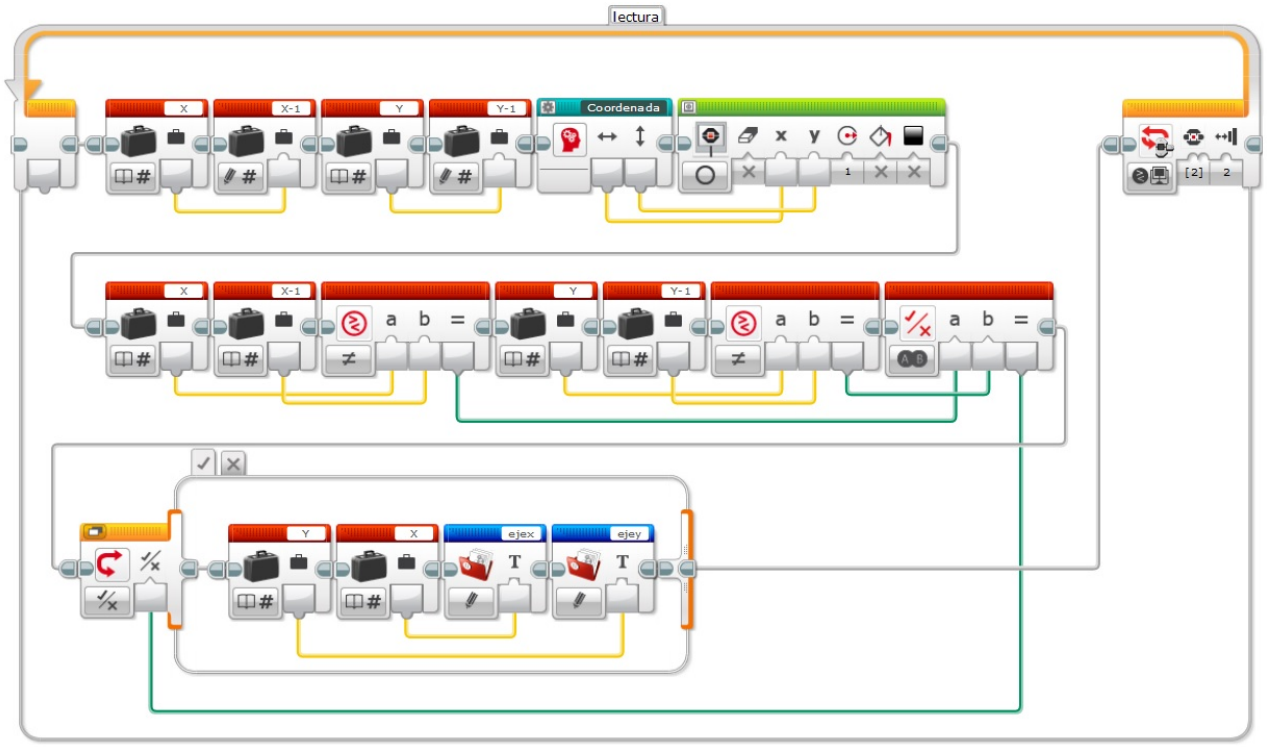


Figura 7.32 – Robot trazador: Bucle 'lectura'

Comienza guardando el valor de «X» en «X-1» y el valor de «Y» en «Y-1», a continuación ejecuta el bloque coordenada con sus salidas conectadas al bloque pantalla, en éste la opción borrar está configurada como falsa, para permitir trazar líneas.

Compara los valores de «X» con «X-1» e «Y» con «Y-1», si en alguno de los dos casos los valores difieren entre sí, guarda los valores de la variables X e Y en sendos archivos. Realizamos esta comprobación para no almacenar datos repetidos (7.32).

7.6.4.0.3. Bucle 'Movimiento' : Comienza reiniciando la lectura del giroscopio. Mediante bloques comparadores y bloques tipo switch anidados, determina en qué dirección debe moverse el robot. Por ejemplo:

- Si «Y» e «Y-1» son iguales, el robot debe moverse en la dirección X.
- Si nos movemos en X y «X» es mayor que «X-1» el robot va hacia la derecha, en caso contrario va hacia la izquierda.
- Cuando «Y» e «Y-1» dejan de ser iguales, sabemos que se ha producido un cambio de dirección: realizando de nuevo la comparación entre ambos sabemos hacia qué lado debemos girar.

La lógica empleada se comprende mejor mediante el gráfico siguiente:

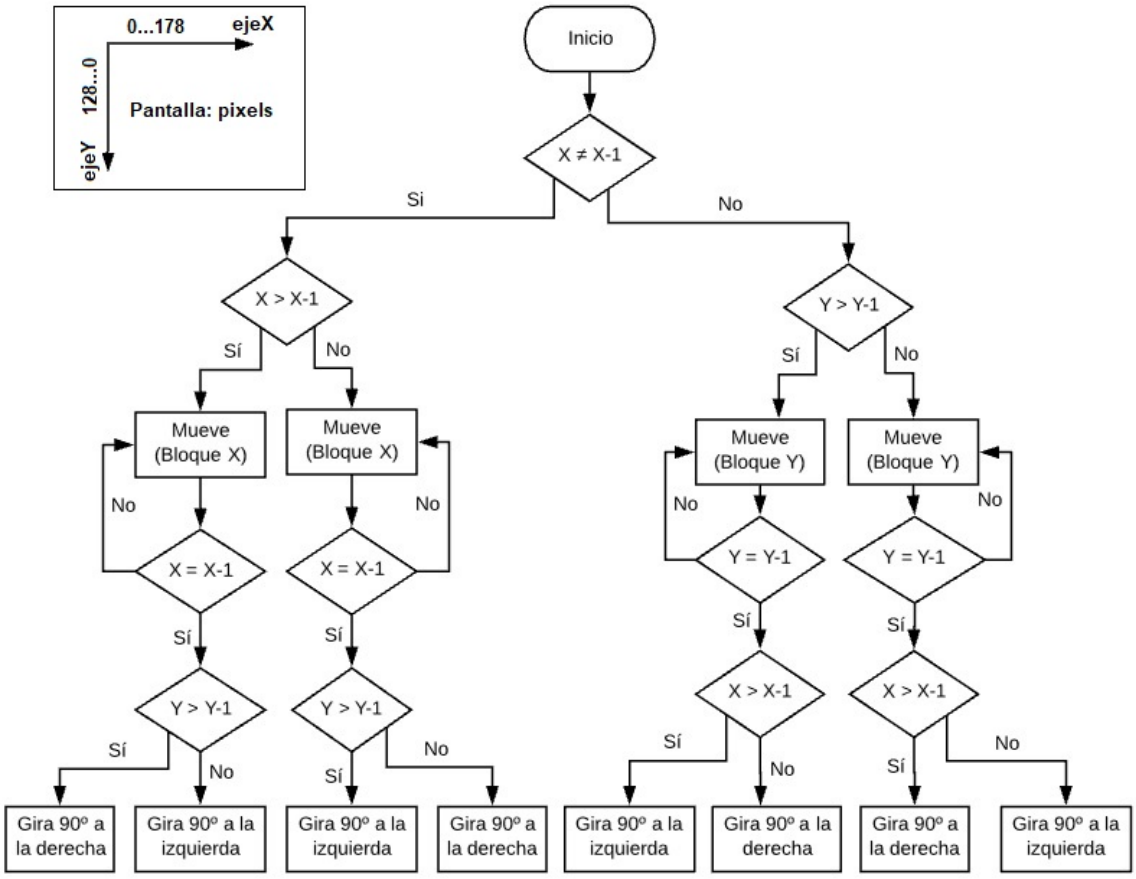


Figura 7.33 – Robot trazador: Bucle 'Movimiento'

7.6.4.0.4. Bloque 'Coordenada' : Es idéntico al utilizado en el ejemplo anterior.

7.6.4.0.5. Bloque 'Inicio' : Comienza reiniciando las lecturas de los encoders C y D y eliminando los archivos «ejex» y «ejey» generados en otras ejecuciones anteriores del programa.

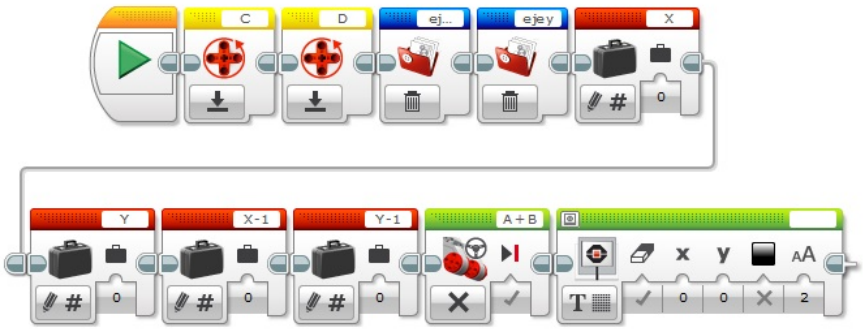


Figura 7.34 – Robot trazador: Bloque 'Inicio'

Da valor 0 a las variables «X», «X-1», «Y» e «Y-1», detiene el bloque formado por los motores A y B y borra la pantalla (7.34).

7.6.4.0.6. Bloque 'MoverX' : Comienza realizando una lectura de «ejex» y «ejey» almacenándola en «X-1» e «Y-1» respectivamente. Acto seguido entra en un bucle cuya condición de salida es de tipo lógico.

Vuelve a realizar otra lectura, esta vez almacenándola en «X» e «Y». A continuación compara los valores de X e X-1, esta condición lógica es la que hace que se mantenga o salga del bucle.

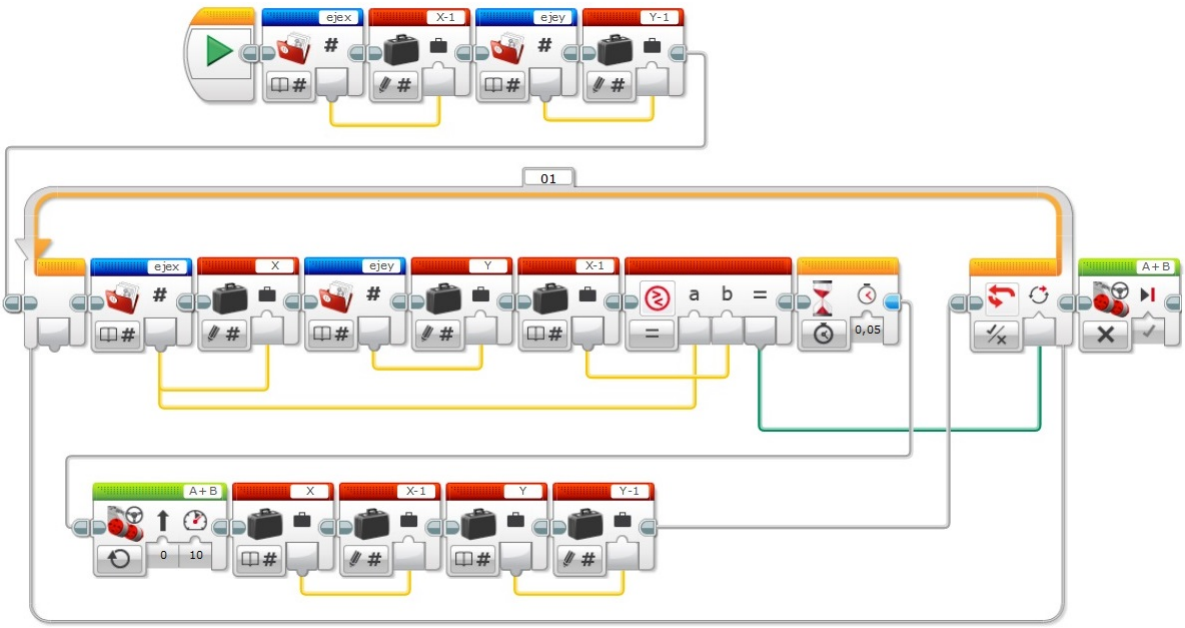


Figura 7.35 – Robot trazador: Bloque 'MoverX'

Vuelca «X» en «X-1» e «Y» en «Y-1» y espera 0.05s hasta la siguiente iteración. Aumentando o disminuyendo este tiempo el robot realizará movimientos más amplios o más cortos. Mientras el programa está dentro del bucle el bloque formado por los motores A y B avanza en línea recta con una potencia de 10, al salir del bucle se detiene (7.35).

7.6.4.0.7. Bloque 'MoverY' : Funciona de manera análoga pero en este caso comparando «Y» e «Y-1».

7.7. Cinta clasificadora

7.7.1. Introducción

En este montaje construiremos una cinta transportadora que clasifique paquetes por tamaño y color mediante el uso de un sensor ultrasónico y un sensor óptico. Dispondremos de paquetes azules y blancos de dos tamaños.

- Los paquetes pequeños (blancos y azules) saldrán al final de la cinta.
- Los paquetes grandes azules son empujados fuera de la cinta por una palanca accionada por un motor mediano, al principio de la cinta, después de los sensores.
- Los paquetes grandes blancos son expulsados más adelante por un vástago conectado a un motor mediano mediante una cremallera.

Dos sensores táctiles funcionarán como botones de marcha/paro y parada de emergencia respectivamente, una vez pulsada la parada de emergencia no se puede volver a arrancar sin reiniciar el programa.

La cinta y los actuadores pueden ajustarse mediante los botones del bloque programable.

A diferencia de la cinta del Ejemplo 1, donde el sensor se ubicaba justo antes del actuador y solo tendríamos dos situaciones posibles (pieza/no pieza), en este caso, los sensores se ubican al principio de la cinta y tendremos varios tipos de piezas, por lo que es necesario "memorizar" qué tipo de pieza ocupa qué lugar. Para ello utilizaremos 3 vectores de tipo lógico para las variables («Azul», «Blanco» y «Grande»), cuyas posiciones iremos desplazando.

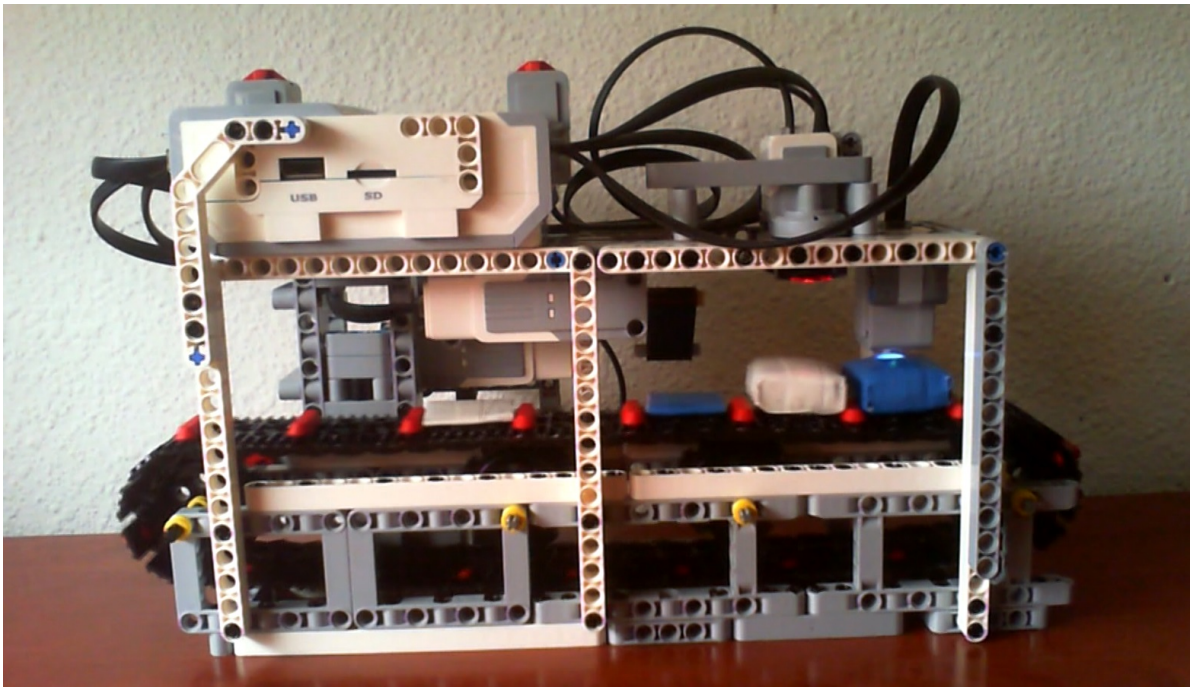


Figura 7.36 – Cinta clasificadora

7.7.2. Sensores y actuadores

Utilizaremos como sensores:

- Un sensor óptico para comprobar el color (Puerto 1).
- Un sensor ultrasónico para comprobar el tamaño (Puerto 4).

Utilizaremos como sensores:

- Un motor grande para mover la cinta (Puerto A).
- Un motor mediano que mueve una palanca (Puerto B).
- Un motor mediano que mueve una cremallera (Puerto C).

7.7.3. Variables

Nombre	Tipo	Descripción
Blanco	Vec. Bool	(0...7), almacena el valor lógico “Blanco” en cada posición de la cinta.
Azul	Vec. Bool	(0...7), almacena el valor lógico “Azul” en cada posición de la cinta.
Grande	Vec. Bool	(0...7), almacena el valor lógico “Grande” en cada posición de la cinta.
Permiso	Bool	Permite o bloquea el funcionamiento de la cinta.

Tabla 7.5 – Cinta clasificadora: Variables

7.7.4. Descripción del programa

El programa ejecuta tres líneas de código simultáneas: una para el funcionamiento normal de la cinta, otra que controla la parada de emergencia y por último un segmento de código que nos permite el ajuste fino de los actuadores (una vez se ha realizado la parada de emergencia).

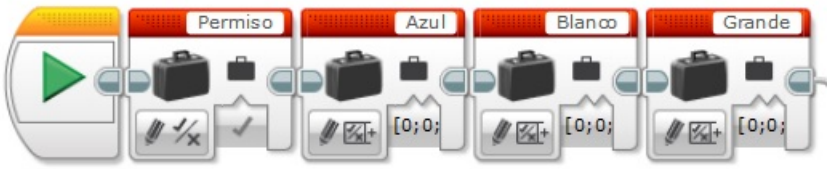


Figura 7.37 – Cinta clasificadora:Vectores lógicos

El segmento ’Funcionamiento’ comienza otorgando valor verdadero a la variable lógica «Permiso», que permite el funcionamiento normal del programa, tras lo cual declara los vectores lógicos «Azul», «Blanco» y «Grande» (7.37).

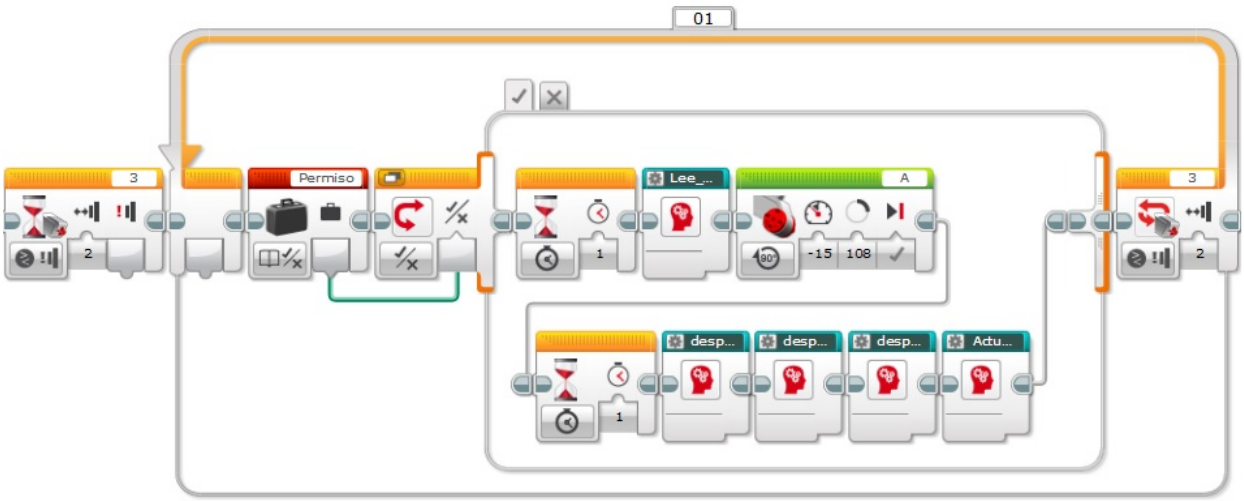


Figura 7.38 – Cinta clasificadora:Bucle funcionamiento

Acto seguido entra en un bucle infinito y espera a que se pulse el sensor táctil (Puerto 3), una vez pulsado entrará en otro bucle que se ejecutará hasta que se vuelva a pulsar (este pulsador funciona como marcha y como paro, 7.38).

Dentro de este bucle, evalúa la variable «Permiso», si esta tiene valor verdadero (lo tendrá salvo el caso en que se haya pulsado la parada de emergencia), espera un segundo y ejecuta el bloque 'Lee sensores', que almacena un valor de tipo bool en la primera posición de los vectores lógicos, tras lo que mueve la cinta 108 grados con una potencia de 15 y espera de nuevo un segundo.

A continuación ejecuta tres bloques que desplazan 1 posición los vectores lógicos «Azul», «Blanco» y «Grande» respectivamente. Por último el bloque 'actuadores' lee el valor de los vectores booleanos y determina si debe activar los actuadores.

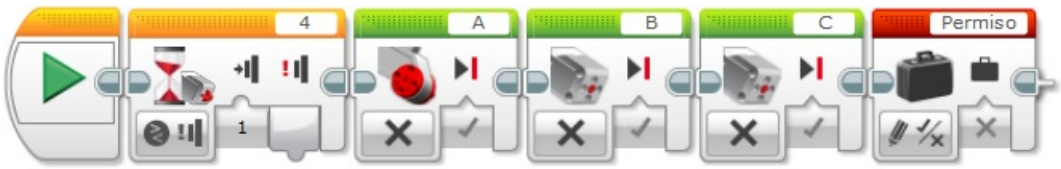


Figura 7.39 – Cinta clasificadora:Parada de emergencia

El segmento que controla la parada de emergencia simplemente espera a que se active el sensor táctil (Puerto 4), una vez pulsado detiene todos los motores y da valor 'falso' a la variable «Permiso» (7.39).

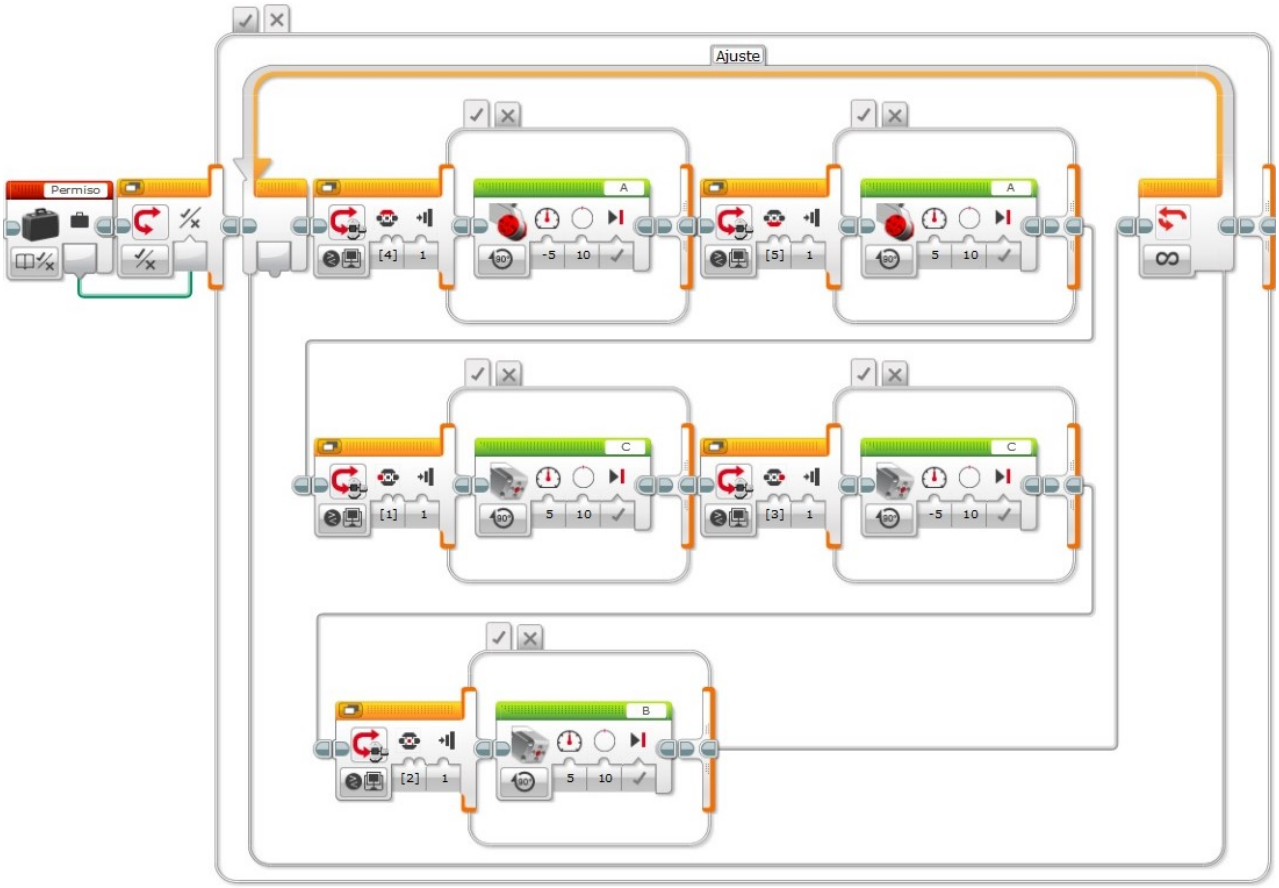


Figura 7.40 – Cinta clasificadora:Ajuste

El segmento 'Ajuste', evalúa la variable «Permiso», si es el valor es 'falso', significa que se ha pulsado parada de emergencia y mediante bloques tipo switch condicionados a la pulsación de los botones del bloque programable mueve los actuadores con una potencia de 5 en incrementos de 10º (7.40).

7.7.4.0.1. Bloque 'Lee sensores' :Mediante un bloque switch, tomando la lectura del sensor óptico en modo detención de color, otorga un valor verdadero o falso al primer elemento del vector «Blanco», en función de que detecte el color blanco o negro (color de la cinta, indicaría que no hay ningún objeto).

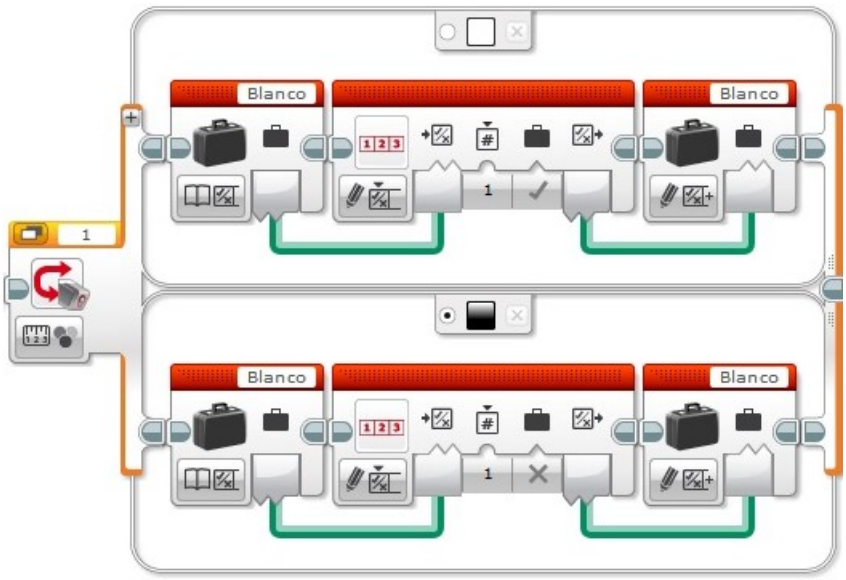


Figura 7.41 – Cinta clasificadora:Bloque 'Lee Sensores'(Detalle)

Realiza la misma operación para el vector «Azul».

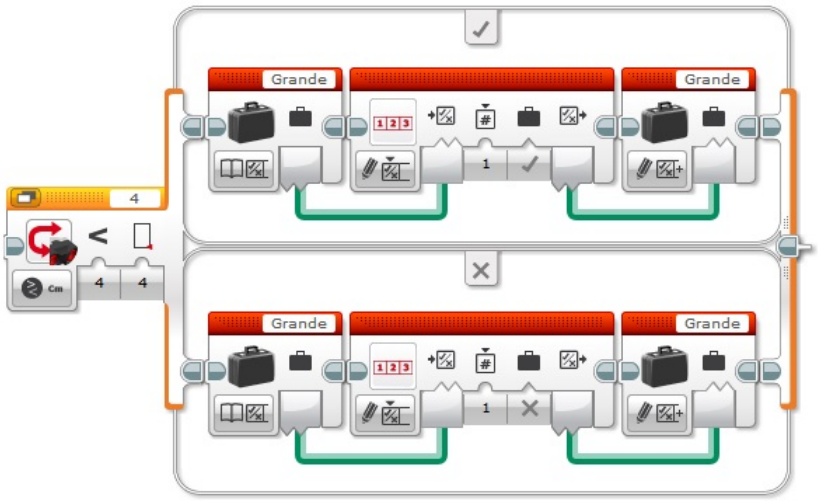


Figura 7.42 – Cinta clasificadora:Bloque 'Lee Sensores'(Detalle)

Para la detección de tamaño utiliza un código similar, esta vez el switch comprueba que la distancia al objeto es menor de 4cm, si es así, clasifica el elemento como grande y da valor verdadero al primer elemento del vector «Grande»(7.42).

7.7.4.0.2. Bloque 'Desplaza' : Este bloque se encarga de desplazar los elementos de los vectores: el elemento de la posición 5 pasa a la 6, el elemento de la posición 4 pasa a la 5 y así sucesivamente. En el caso de que fuese un vector de muchos elementos sería conveniente programar un bucle.

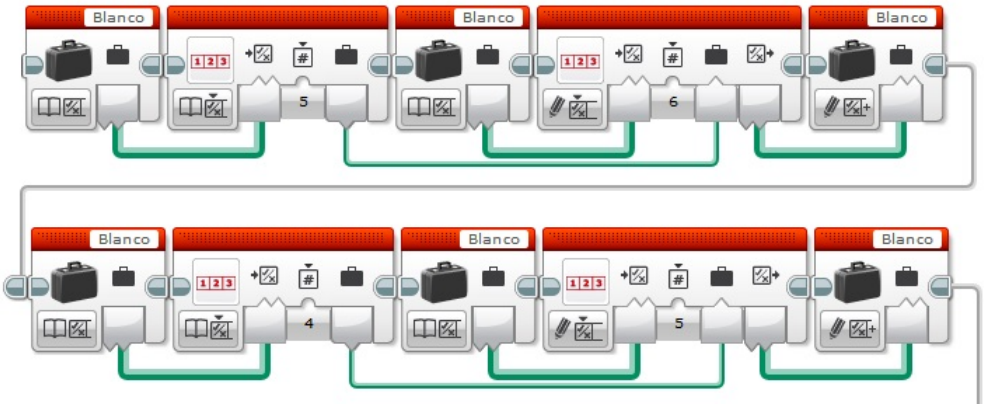


Figura 7.43 – Cinta clasificadora:Bloque 'Desplaza'(Detalle)

Utilizamos un bloque de este tipo para cada uno de los tres vectores («Blanco», «Azul» y «Grande»).

7.7.4.0.3. Bloque 'Actuadores' : Gobierna los motores en función del contenido de las posiciones de los vectores lógicos.

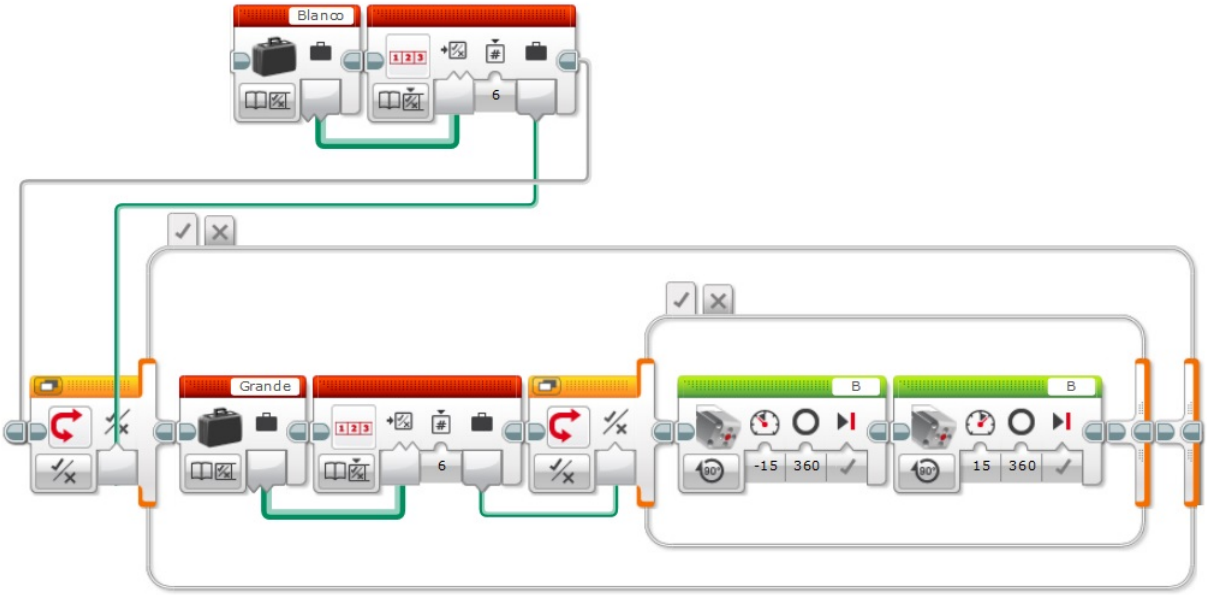


Figura 7.44 – Cinta clasificadora:Bloque 'Actuadores'(Detalle)

Comienza evaluando la posición 6 del vector «Blanco», si la pieza en esta posición es blanca y además es de tipo grande, expande y contrae el vástago conectado al motor B (7.44).

Posteriormente realiza una operación similar evaluando la posición 3 del vector «Azul» y actuando sobre el motor conectado al puerto C.

7.8. Robot Segway

7.8.1. Introducción

El objetivo del siguiente montaje es la realización de un robot seguidor de línea que se desplace manteniendo el equilibrio sobre dos ruedas. Para ello combinaremos el algoritmo PID implementado anteriormente, con un algoritmo de control del equilibrio basado en el del robot 'Gyro Boy' (incluido en el set educativo LEGO) sobre una base motriz de diseño propio.

Debido a la complejidad del programa y a que LEGO, no proporciona una explicación detallada de su funcionamiento, el análisis del algoritmo de control, supone un ejercicio interesante de "ingeniería inversa" que nos permite ahondar en las posibilidades de la programación EV3.

Las variables de entrada al sistema son:

- Velocidad angular del robot (grados/s): medida directamente por el giroscopio.
- Ángulo del robot: se obtiene integrando la velocidad angular.
- Posición del motor (grados): medida directamente por el encoder.
- Velocidad del motor: se obtiene derivando la posición del motor.

Como variable de salida se genera una variable numérica correspondiente a la potencia a entregar al par de motores.

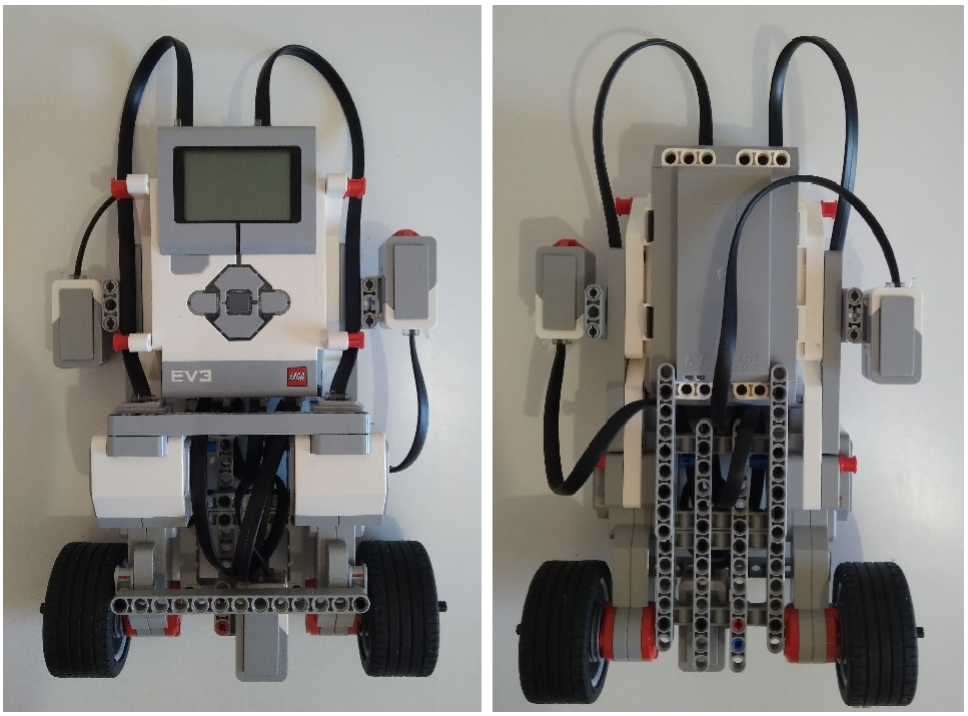


Figura 7.45 – Robot Segway

7.8.2. Sensores y actuadores

Utilizaremos como sensores:

- Un sensor táctil a modo de pulsador (Puerto 4).
- Un sensor giroscópico para la velocidad angular del robot (grados/s). (Puerto 2).
- Un sensor óptico para el seguimiento de línea. (Puerto 3).
- Encoders integrados en los motores para medir la posición (en grados) de las ruedas.

Como actuadores utilizaremos dos motores grandes (Puertos A y D).

7.8.3. Variables

Nombre	Tipo	Descripción
mSuma	Num	Valor de la suma de la lectura de los encoders (grados).
mPos	Num	Valor acumulado de la variable mDP.
mDP	Num	Diferencia el valor actual de mSuma y el valor en t-1.
mDP1	Num	Valor de mDP en t-1.
mDP2	Num	Valor de mDP en t-2.
mDP3	Num	Valor de mDP en t-3.
Direc	Num	Parámetro que gobierna el giro del robot.
Avanc	Num	Parámetro que gobierna el avance del robot.
Iter	Num	Contador de bucle.
gAng	Num	Ángulo del robot respecto a la vertical (en grados).
ptncia	Num	Potencia entregada al bloque de motores.
est	Num	Variable de estado (toma valores 0 y 1).
gMin	Num	Valor mínimo de la aceleración angular (grados/s).
gMax	Num	Valor máximo de la aceleración angular (grados/s).
gSuma	Num	Valor acumulado de la aceleración angular (grados/s).
giro	Num	Lectura del giroscopio (grados/s).
goff	Num	Valor medio del offset del giroscopio (grados/s)
t_int	Num	Tiempo medio de iteración de bucle.
gVel	Num	Valor corregido de la velocidad angular del robot (grados/s).
mVel	Num	Velocidad angular del motor (grados/s).
sale	Bool	Condición de salida del bucle equilibrio.

Tabla 7.6 – Robot Segway: Variables

7.8.4. Descripción del programa

7.8.4.0.1. Bucle 'Giroscopio' : Gobierna el equilibrio del robot.

Figura 7.46 – Robot Segway: Bucle 'Giroscopio' (detalle)

Con el robot en posición vertical, pulsamos el sensor táctil, el programa ejecuta el bloque 'Inicio', que reinicia variables y lecturas de encoders y giroscopio, así como temporizadores, el robot muestra el mensaje *CALIBRANDO* en la pantalla.

Acto seguido entra en el bloque 'gOffset', que comprueba que el robot está inmóvil y calcula el offset en la medida del giroscopio. Una vez hecha esta comprobación, el robot emite un sonido y muestra el mensaje *EQUILIBRIO*, la variable «est» toma valor 1, permitiendo que el programa entre en el bucle 'equlibrio' (7.46).

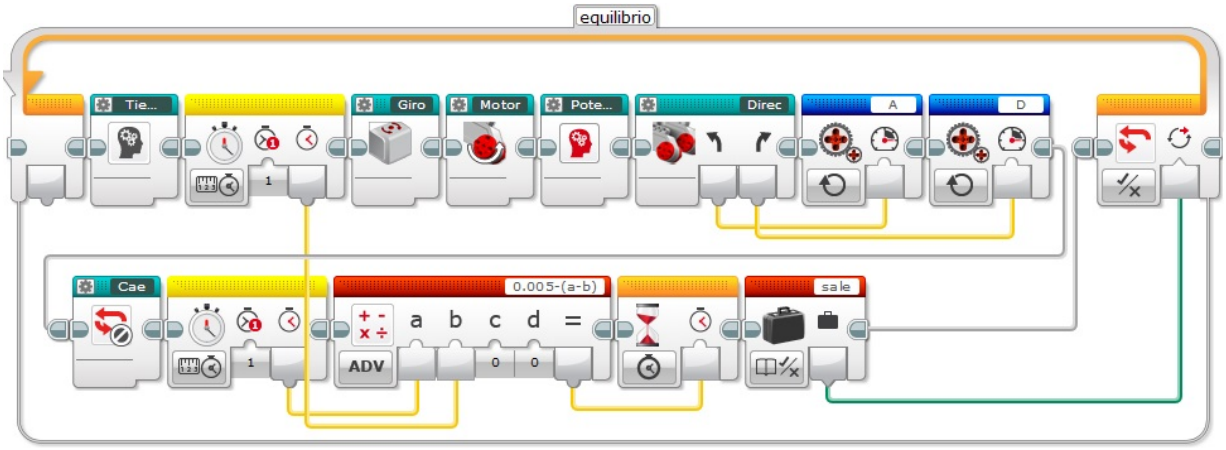


Figura 7.47 – Robot Segway: Bucle 'equilibrio'

Una vez dentro de dicho bucle, lo primero en ejecutarse es el bloque 'Tiempo', que nos proporciona el tiempo promedio de iteración del bucle, «t_int», que nos servirá para el cálculo del ángulo de inclinación y de la velocidad de los motores.

A continuación se sitúan los bloques de lectura del giroscopio y los motores. El bloque 'Giro', nos proporciona los valores de velocidad angular y ángulo del robot, mientras que el bloque 'Motor' nos proporciona la medida de posición de las ruedas y la velocidad de las mismas.

El bloque 'Potencia' utiliza la ecuación diseñada por LEGO en su robot *Gyro Boy*. Suma los parámetros leídos de los dos bloques anteriores además del parámetro «Avanc», cada uno multiplicado por un factor, para ponderar sus pesos en dicha ecuación.

El bloque 'Cae' comprueba si las ruedas dejan de tener contacto con el suelo (el robot ha caído o lo hemos levantado del suelo), en cuyo caso activa una variable de tipo booleano que usamos para interrumpir el bucle.

El bloque matemático que se incluye en el bucle sirve para mantener estable el tiempo de iteración del mismo, restando de 5ms el tiempo que tarda en realizar los cálculos, con lo cual el periodo de muestreo se mantiene constante (5ms) independientemente de que el tiempo de cálculo varíe de una iteración a otra (7.47).

Figura 7.48 – Robot Segway: Bucle 'Giroscopio' (detalle)

Cuando sale del bucle 'equilibrio' y vuelve al bucle 'Giroscopio', se paran los motores A y D y la variable «est» (estado) vuelve a 0. Además el bloque emite un pulso de luz roja, muestra el mensaje *ERROR!!!* y emite un sonido a modo de alarma (7.48).

7.8.4.0.2. Bucle 'PID' : Comienza con un bloque de configuración donde podemos configurar el Set Point y las ganancias Kp, Kd y Ki, para el algoritmo PID.

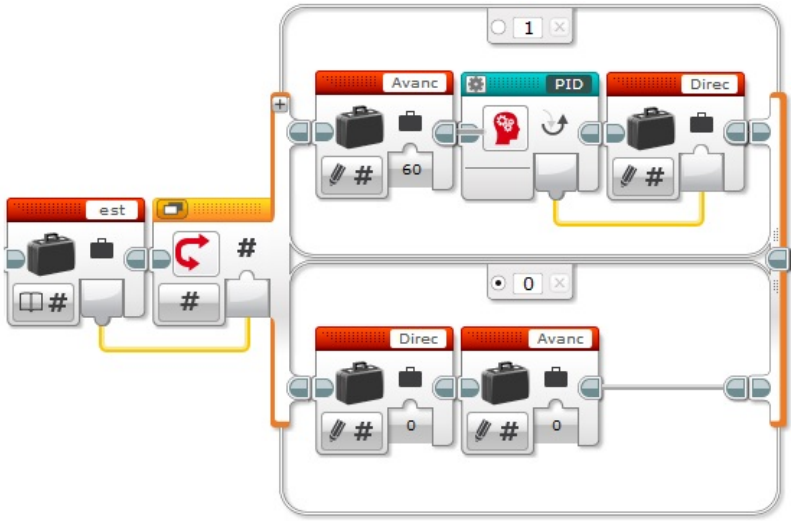


Figura 7.49 – Robot Segway: Bucle 'PID'

Un bucle infinito contiene un switch que evalúa la variable «est», en caso de ser 0, las variables «Avanc» y «Direc» toman valor 0, con lo cual el robot está en reposo. En el caso contrario, la variable «Avanc» toma un valor fijo y la variable «Direc» toma el valor de salida del bloque PID (7.49).

7.8.4.0.3. Descripción detallada de los bloques :

- Inicio: El bloque 'Inicio' comienza reseteando las lecturas de los dos encoders ,el giroscopio y el temporizador 2, que es el utilizado en el bloque que verifica si el robot se ha caído

(bloque 'Cae'). Posteriormente resetea a 0 las variables numéricas y a *falso* la variable booleana que controla la salida del bucle de equilibrio. La variable «gAng» se inicia a un valor que representa el ángulo aproximado que tendrá el robot en su posición de reposo sobre el soporte que utilicemos antes de ponerlo en funcionamiento.

- gOffset: El bloque 'gOffset' calcula el offset en la lectura del giroscopio. Incluso estando completamente estático el sensor puede entregar una medida diferente de 0 (grados/s), el programa determina este parámetro y lo utiliza posteriormente en los cálculos como corrección. El robot ha de estar completamente parado para poder realizar esta operación.

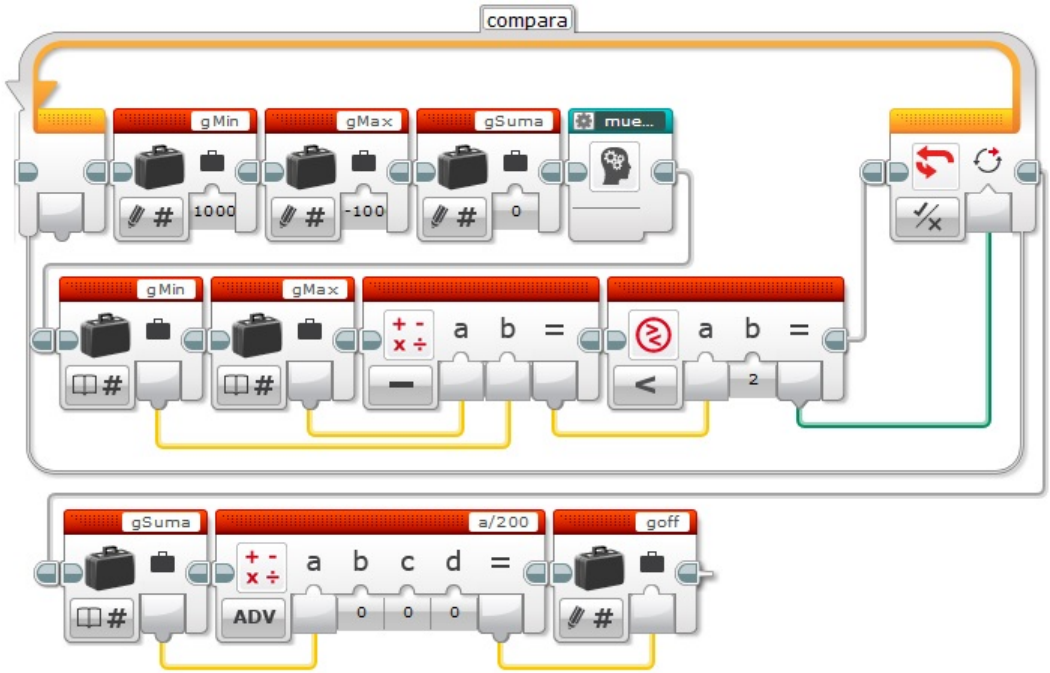


Figura 7.50 – Robot Segway: Bloque 'gOffset'

Las variables «gMin» y «gMax» almacenan los valores mínimo y máximo medidos por el giroscopio (en grados/s) y se inicializan en valores fuera de rango para que se actualicen posteriormente dentro de la subtarea 'muestrea'. La variable «gSuma» se inicializa en 0 y se utiliza para almacenar la suma de las lecturas del giroscopio.

El bloque 'muestrea' es un bucle que se repite 200 veces con una espera de 4ms entre iteraciones. Comienza actualizando el valor de «gSuma» y posteriormente comparamos las variables «gMin» y «gMax» con la lectura actual del giroscopio (variable «giro»), sobrescribiéndolas si es necesario.

Al salir de la subtarea 'muestrea', se compara la diferencia entre el valor «gMin» y «gMax», si es mayor de 2 (grados/s) significa que el robot no estaba en reposo con lo que vuelve a ejecutarse el bucle 'compara'. Por el contrario, si la diferencia es menor de 2, divide el valor «gSuma» entre 200 para hallar el valor medio de offset («gOff»), 7.50.

- Tiempo: Lo primero que encontramos dentro del bucle 'equilibrio' es el bloque 'tiempo', este bloque es el primero en ejecutarse y nos proporciona la variable «t_int», que es el tiempo promedio de iteración. Utilizamos el tiempo promedio, ya que el tiempo de ejecución del programa puede variar de una iteración a otra y nos interesa obtener un valor estable.

  Necesitamos este dato de tiempo para obtener la inclinación del robot, ya que la medida que obtenemos del giroscopio esta en grados/s. A su vez, los encoders miden la posición en grados, por lo que utilizaremos este tiempo para hallar la velocidad.

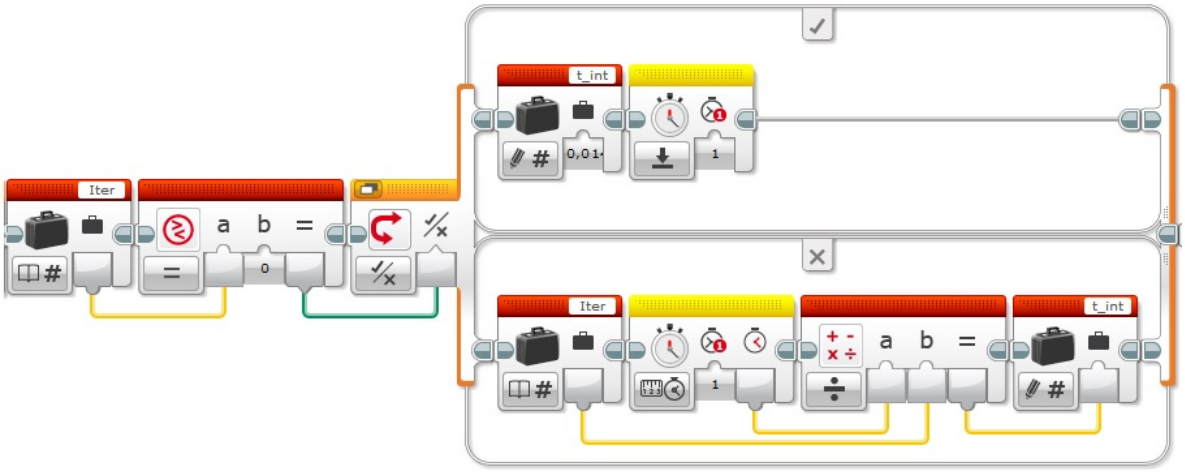


Figura 7.51 – Robot Segway: Bloque 'Tiempo' (Detalle)

La variable «Iter» funciona como contador del bucle y se resetea en el bloque 'Inicio'. El programa contiene un bloque tipo switch lógico que evalúa si dicha variable es 0. Después del switch se incrementa la variable «Iter» en 1, de esta forma, las siguientes veces la condición será falsa.

En la primera iteración «Iter» es efectivamente 0, se asigna un tiempo fijo a «t_int» de 0.014s y se reinicia el temporizador 1. Las siguientes iteraciones, «Iter» es distinto de 0, «t_int» se calcula dividiendo el valor de tiempo del temporizador entre el valor de «Iter», hallando así el valor medio(7.51).

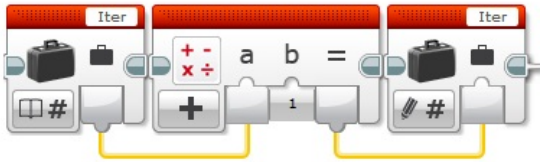


Figura 7.52 – Robot Segway: Bloque 'Tiempo' (Detalle)

Por último se incrementa el valor de «Iter» (7.52).

- Giro: Se calcula un nuevo offset teniendo en cuenta el 99.5 % del valor anterior de «gOff» y el 0.05 % del valor de lectura del giroscopio (g/s). Esta operación se realiza para tener en cuenta la variación de este parámetro en el tiempo.

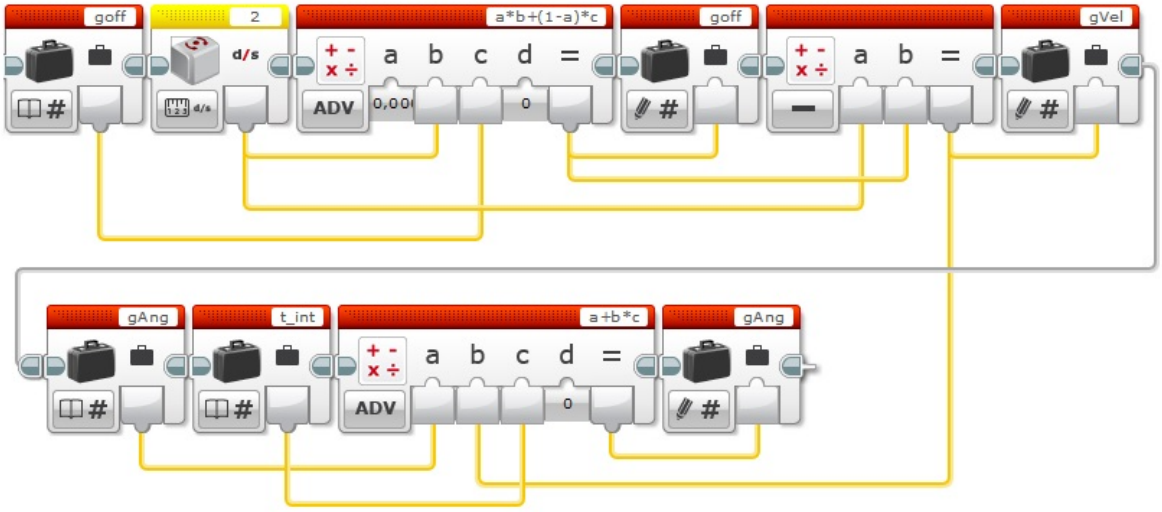


Figura 7.53 – Robot Segway: Bloque 'Giro'

Este valor de offset es el que se resta de la lectura del sensor y se vuelca en la variable «gVel». Mediante la siguiente operación obtenemos el valor de inclinación que almacenamos en «gAng»:

$$gAng = gVel \times t\_int \tag{7.1}$$

- Motor: Este bloque lee los encoders y calcula los valores de posición y velocidad de los motores.

  De manera análoga a como en el anterior bloque multiplicábamos el valor de velocidad angular por el tiempo para obtener el ángulo, en este bloque realizaremos el proceso inverso, dividiremos los valores del ángulo de posición del motor entre el tiempo para obtener la velocidad angular.

  El programa lee los encoders A y D y suma ambos valores almacenándolos en la variable «mSuma», podemos realizar esta operación, ya que cuando el robot gira, los motores giran en sentidos contrarios, con lo cual la suma de A y D se anula, no afectando al cálculo.

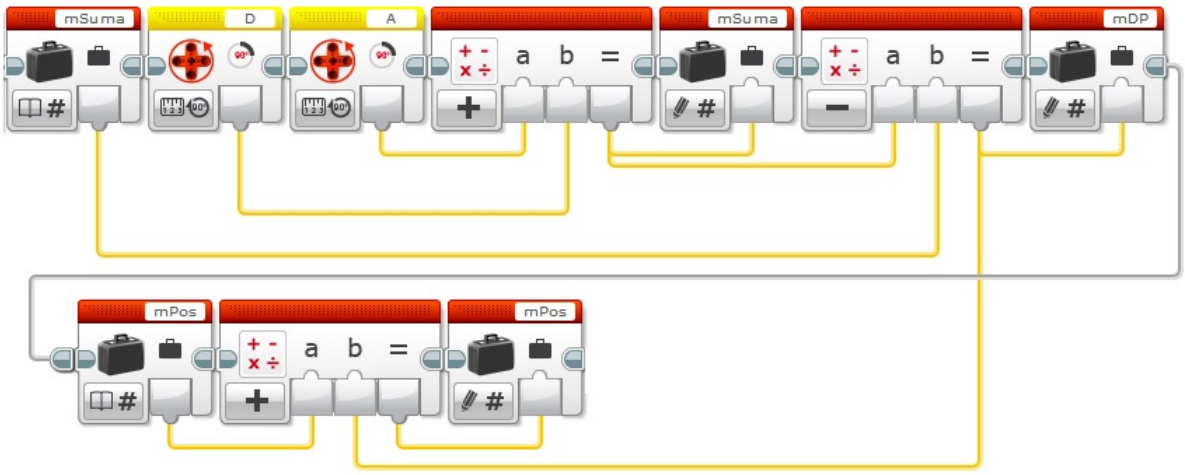


Figura 7.54 – Robot Segway: Bloque 'Motor' (Detalle)

A continuación se calcula la diferencia entre el valor actual de «mSuma» y su valor en la iteración anterior, obteniendo así el valor en grados del giro en la última iteración, almacenamos este dato en la variable «mDP».
El valor acumulado de «mDP» nos da el valor absoluto de posición («mPos»), 7.54.

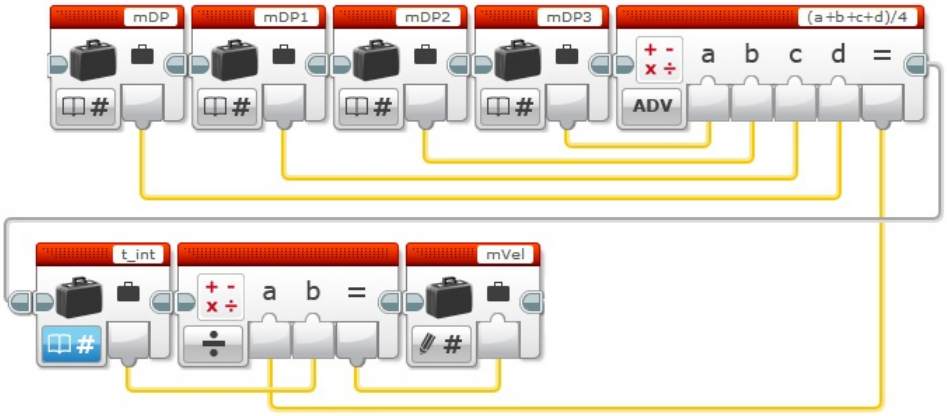


Figura 7.55 – Robot Segway: Bloque 'Motor' (Detalle)

El valor de velocidad «mVel» se obtiene dividiendo la media del valor actual «mDP» y los tres últimos valores (almacenados respectivamente en «mDP1», «mDP2», «mDP3») entre el valor de tiempo «t_int» (7.55).

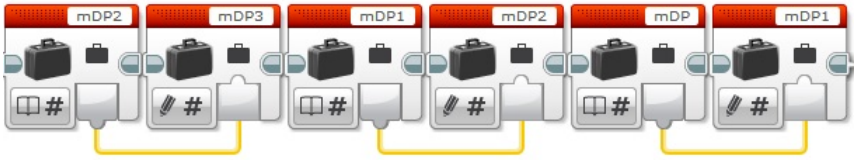


Figura 7.56 – Robot Segway: Bloque 'Motor' (Detalle)

Por último se actualizan las variables (7.56):

mDP3=mDP2, mDP2=mDP1, mDP1=mDP,

- Potencia: regula la cifra de potencia que se envía a los motores A y D.

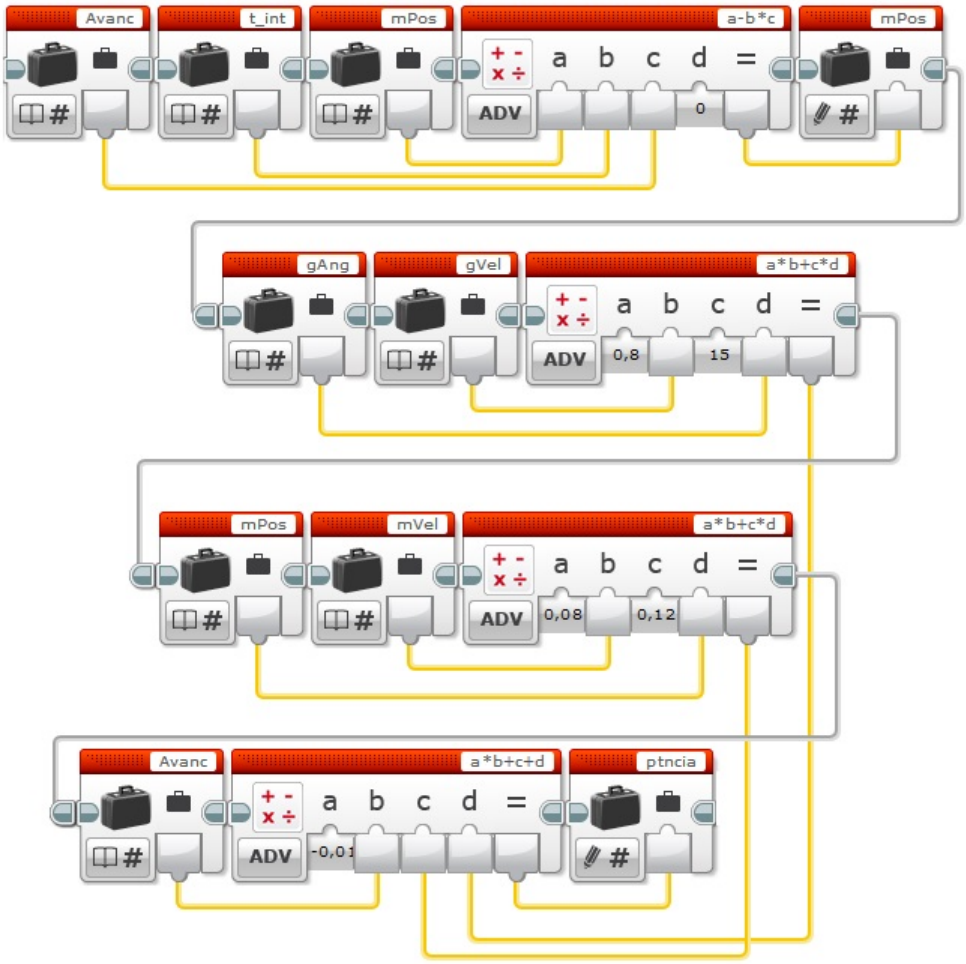


Figura 7.57 – Robot Segway: Bloque 'Potencia' (Detalle)

En la primera parte del programa, la variable «mPos» se actualiza. Primero multiplicamos la variable «Avanc» por «t_int» (distancia = velocidad * tiempo), el resultado de esta operación se resta al valor anterior de «mPos», devolviendo un nuevo valor para la posición del motor.

Por último se aplica la ecuación que pondera los pesos de las variables «gAng», «gVel», «mPos», «mVel» y «Avanc». Y que genera la acción de control (variable «ptncia»), 7.57:

$$ptncia = K1.Avanc + K2.gAng + K3.gVel + k4.mPos + K5.mVel \tag{7.2}$$

Las ganancias utilizadas por LEGO y que condicionan el funcionamiento del robot son:

- K1= -0.01
- K2= 15
- K3= 0.8
- K4= 0.12
- K5= 0.08

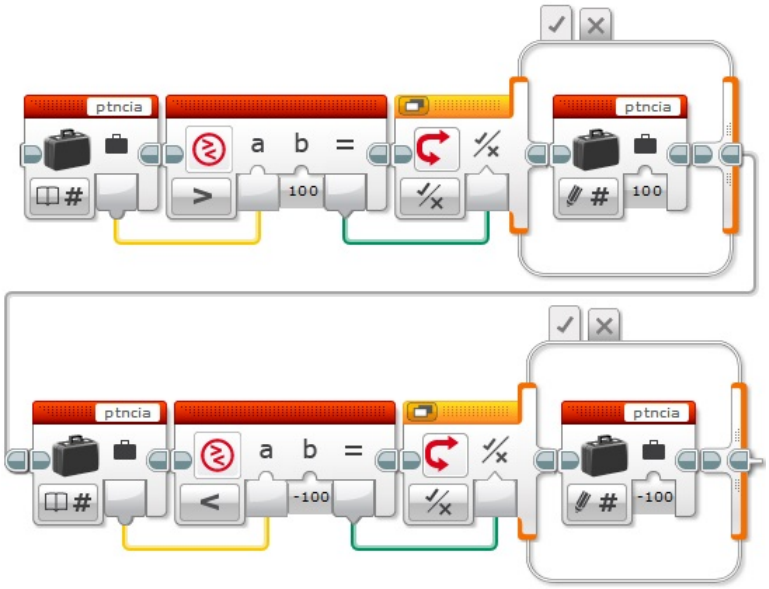


Figura 7.58 – Robot Segway: Bloque 'Potencia'(Detalle)

Por último se ajusta el rango de la variable de control, ya que el rango de valores aceptado por los motores LEGO es (-100,100), 7.58.

- Direc

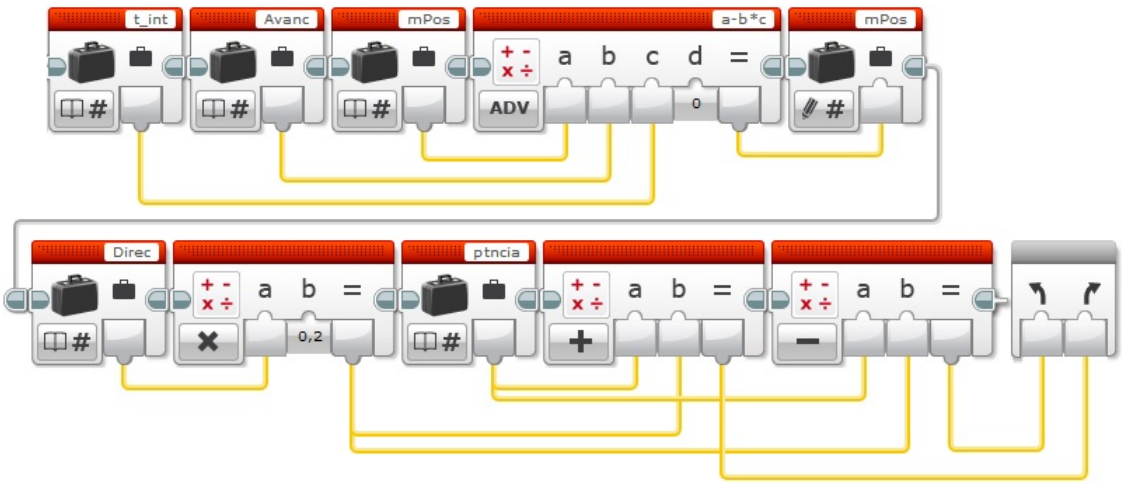


Figura 7.59 – Robot Segway: Bloque 'Direc'

El bloque comienza actualizando de nuevo la variable «mPos» de manera análoga a como se hizo en el bloque anterior.

Posteriormente utiliza la variable «ptncia» generada anteriormente y la combina con la variable «Direc», para generar la diferencia de potencia entre el motor derecho e izquierdo que hace que el robot gire.
La variable «Direc», se multiplica por un factor de 0.2, modificando este factor, podemos hacer que el robot realice giros más cerrados, generando una diferencia mayor en el movimiento de las ruedas.

- Cae: Este bloque evalúa si el robot está en equilibrio o ha caído.

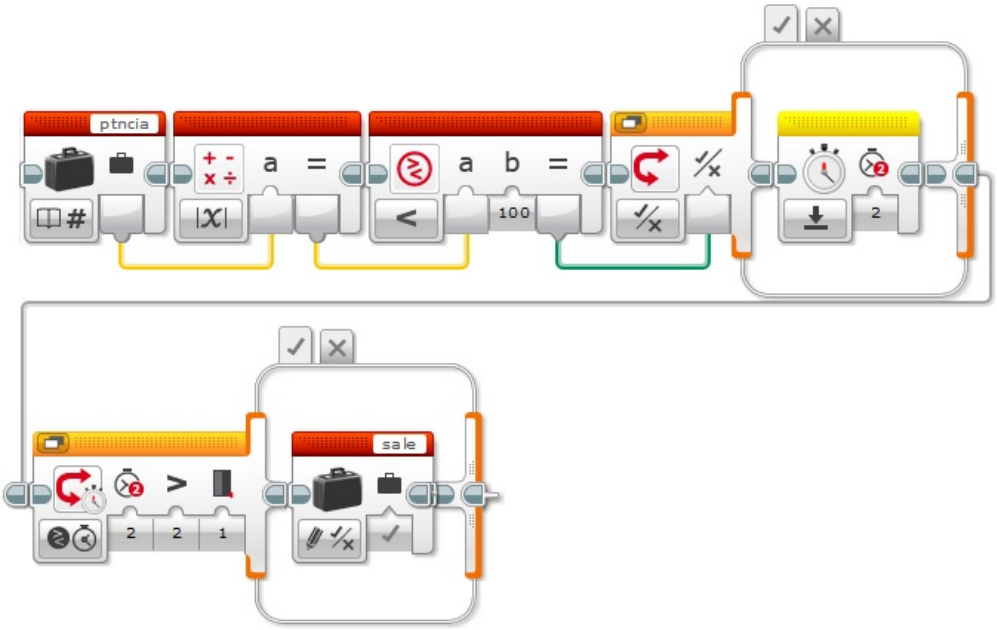


Figura 7.60 – Robot Segway: Bloque 'Cae'

El primer switch comprueba si el valor absoluto de la variable «ptncia» es menor de 100, si es así, se reinicia el temporizador 2, en caso contrario no se efectúa ninguna acción.

El segundo switch evalúa la cuenta del temporizador, si es menor de 1, entonces la variable lógica sale continúa a 0 (se inicializa a 0 en el bloque Inicio) permitiendo que el bucle equilibrio se siga ejecutando.

Por el contrario, si el primer switch detecta que «ptncia» tiene un valor absoluto de 100 (no puede ser mayor, ya que lo limitamos en el bloque 'Potencia') el temporizador 2 no se reinicia, con lo que cuando la lectura del temporizador sea mayor de 1 en el segundo switch, la variable sale toma valor 1, haciendo que el programa salga del bucle de equilibrio (7.60).

- Conf: Este bloque simplemente nos permite, por comodidad, introducir como entradas directamente los valores de Set Point, Kp, Ki y Kd que utilizaremos en el bloque PID.

- PID: Este bloque contiene el algoritmo PID que actúa sobre el parámetro «Direc», que gobierna el giro del robot. El algoritmo es exactamente igual al utilizado en el seguidor de línea implementado anteriormente y explicado en el apartado correspondiente.

7.9. Telégrafo

7.9.1. Introducción

En este montaje construiremos un telégrafo. Mediante un pulsador teclearemos código morse que aparecerá en la pantalla. Un segundo pulsador nos permitirá introducir un espacio y mediante la botonera del brick, podremos borrar la última letra, borrar todo o efectuar un salto de línea. El salto de línea se efectuará también de forma automática cuando hayamos completado una línea completa de caracteres.

A	·−	J	·−−−	S	···
B	−···	K	−·−	T	−
C	−·−·	L	·−··	U	··−
D	−··	M	−−	V	···−
E	·	N	−·	W	·−−
F	··−·	O	−−−	X	−··−
G	−−·	P	·−−·	Y	−·−−
H	····	Q	−−·−	Z	−−··
I	··	R	·−·		

Figura 7.61 – Abecedario en código Morse

Al mismo tiempo el texto se irá guardando en un archivo, añadiendo una nueva línea al archivo cada vez que ejecutemos un salto de línea. La opción «Borrar Todo», borrará también el archivo, además de la pantalla.

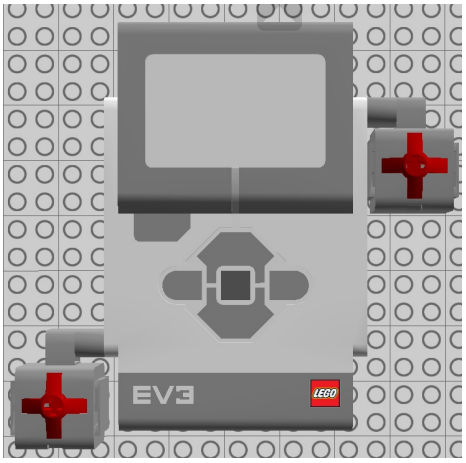


Figura 7.62 – Modelo 3D del robot telégrafo

7.9.2. Sensores y actuadores

Utilizaremos dos sensores táctiles (Puertos 1 y 2) para introducir las letras en código morse y el espacio respectivamente.

7.9.3. Variables

Nombre	Tipo	Descripción
Indice	Num.	Indica el número de elementos del vector «Letra».
LeePulso	Bool	Indica cuando se ha acabado de leer una letra en morse.
Puntoraya	Num.	0=punto, 1=raya.
Letra	Vec. Num.	Contiene la letra en código morse.
Caract	Texto	Contiene la letra en formato texto.
AntCar	Texto	Contiene la concatenación de letras en formato texto.
Espacio	Texto	Espacio en blanco.
Contador	Num	Cuenta el número de caracteres por línea de texto.
Y	Num	Coordenada Y para impresión en pantalla.
N_linea	Num	Número de línea.
Anterior	Texto	Guarda la concatenación de texto anterior.

Tabla 7.7 – Telégrafo: Variables

7.9.4. Descripción del programa

Comienza eliminando el archivo de texto correspondiente a ejecuciones anteriores del programa, posteriormente ejecuta el bloque 'Inicio', este bloque borra la pantalla y pone a cero las variables numéricas y borra la cadena de caracteres que almacena la línea de texto.

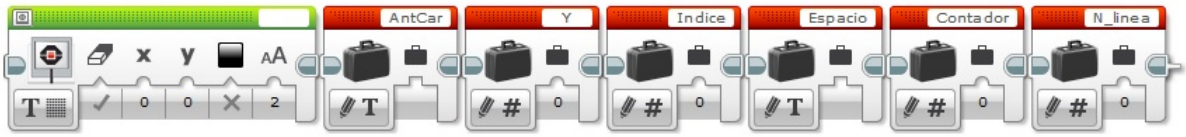


Figura 7.63 – Telégrafo: Bloque Inicio

A continuación se bifurca en cinco segmentos, cada uno formado por un bucle.

7.9.4.0.1. Bucle 'Lee' : Se trata de un bucle infinito. El bucle comienza con un bloque de espera condicionado a la pulsación del pulsador 1, mientras se mantiene pulsado emite un pitido y se reinicia el temporizador 1, la lectura de este temporizador será la que utilicemos para determinar si la pulsación se corresponde con un punto o una raya.

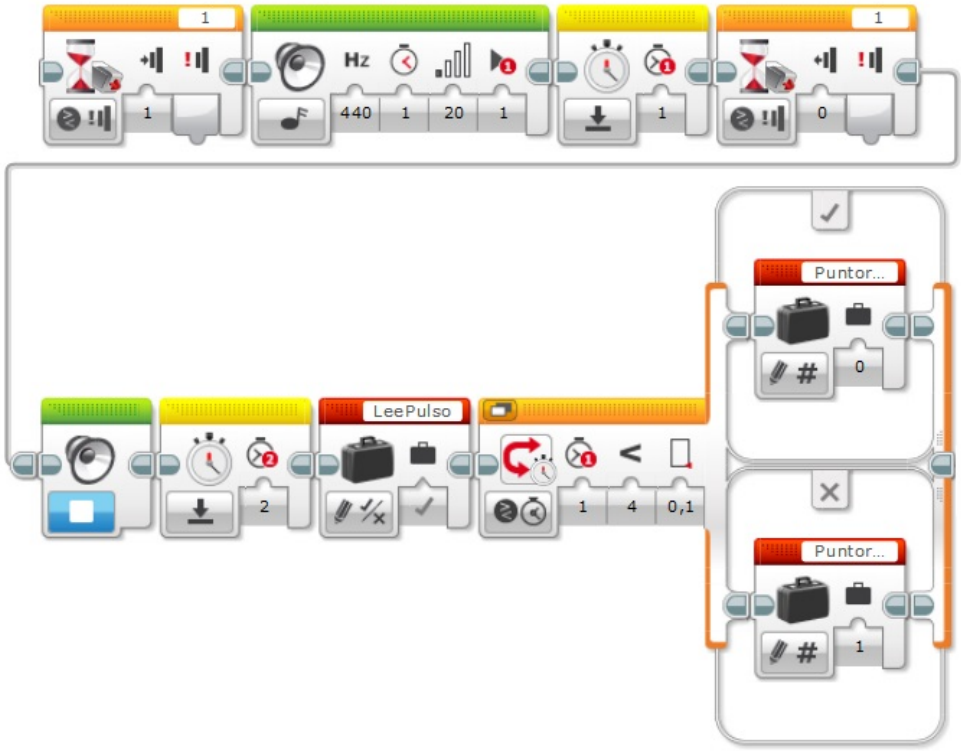


Figura 7.64 – Telégrafo: Bucle 'Lee'(Detalle)

Al soltar el pulsador, cesa el pitido y se reinicia el temporizador 2, este temporizador mide el tiempo entre pulsaciones, acto seguido da valor verdadero a la variable lógica «LeePulso». La lectura del temporizador 2 y la variable «LeePulso» se utilizarán en el siguiente bucle.

A continuación un bloque tipo switch toma la lectura del pulsador 1. Si la duración de la pulsación es menor de 0.1 s, da valor 0 (punto) a la variable numérica «Puntoraya», de lo contrario da valor 1 (raya), 7.64.

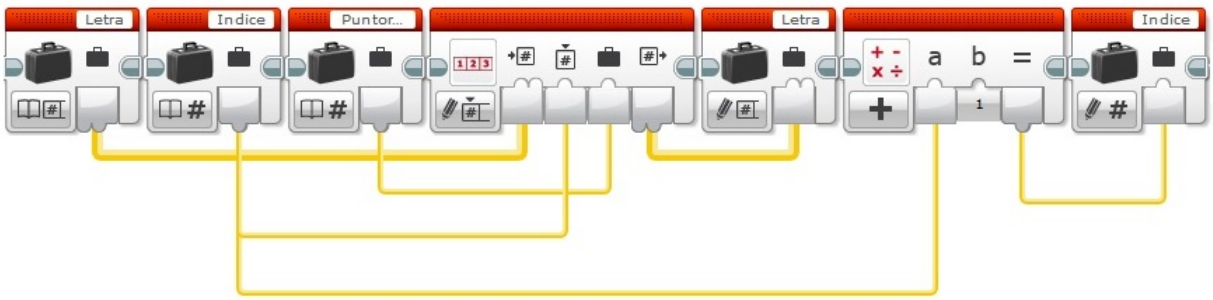


Figura 7.65 – Telégrafo: Bucle 'Lee' (Detalle)

Acto seguido toma el valor de «Puntoraya» y lo introduce en el vector numérico «Letra» en el índice correspondiente, el valor de «Indice» comienza siempre en 0 y se incrementa en cada iteración de bucle (7.65).

7.9.4.0.2. Bucle 'Procesa' : Se trata de nuevo de un bucle de tipo infinito, evalúa la variable lógica «LeePulso».

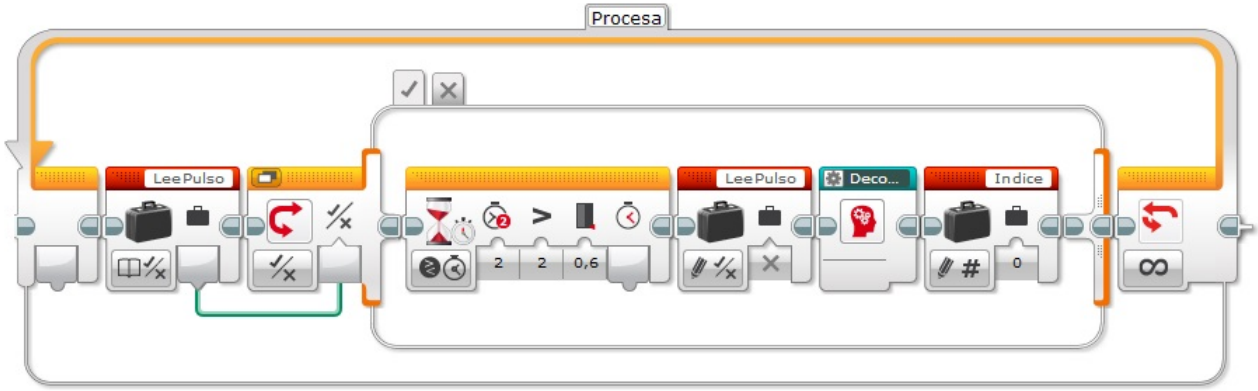


Figura 7.66 – Telégrafo: Bucle 'Procesa'

Si «LeePulso» tiene valor verdadero y el valor del temporizador 2 es mayor a 0.6 s da valor falso a «LeePulso» y ejecuta el bloque decodifica. Esto quiere decir que una espera mayor de 0.6 s entre pulsaciones significa que se ha acabado de introducir una letra (7.66).

Dentro de este bucle se incluyen los siguientes subprogramas:

- Bloque 'Decodifica': Mediante un sistema de bloques tipo switch compara los valores (0 o 1) de cada posición del vector numérico «Letra», para determinar de qué letra se trata y asignarle a la variable texto «Caract» dicha letra (Ver sección Planos).

- Bloque 'Anexar': Este bloque es el encargado de formar la cadena de caracteres que compone cada línea de texto mediante el uso del bloque para concatenar texto.

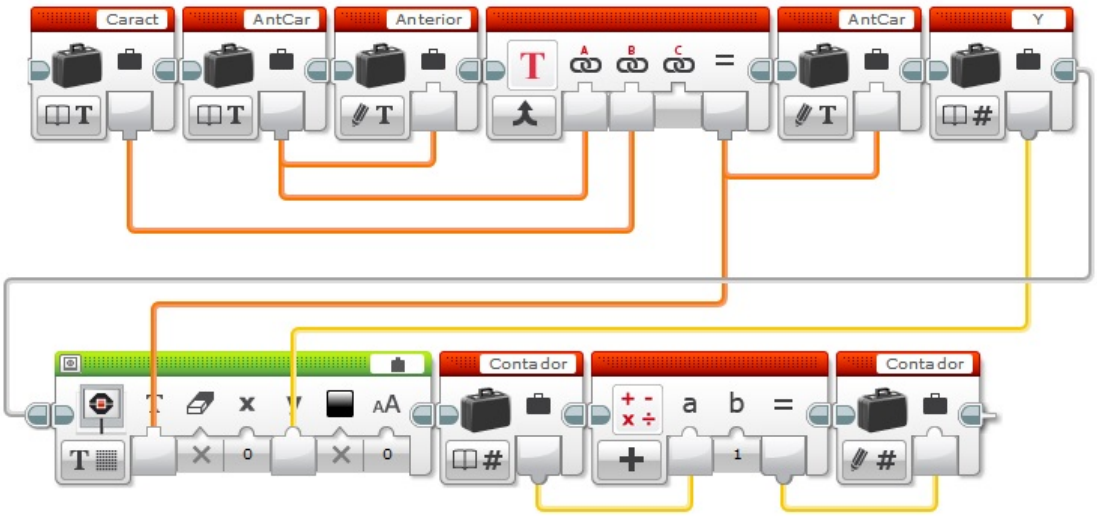


Figura 7.67 – Telégrafo: Bloque 'Anexar'

Une la letra obtenida del bloque 'Decodifica' («Caract») con la cadena de caracteres «AntCar», inicialmente vacía y vuelca el resultado en «AntCar».

La variable «Anterior» guarda el contenido de la cadena de caracteres en la ejecución anterior del bloque, esto servirá para borrar en caso de que nos equivoquemos al introducir la letra.

Un bloque 'Pantalla' imprimirá en pantalla el contenido de «AntCar» con la coordenada Y correspondiente. «Y» es un valor que se incrementa con cada salto de línea.

Por último se incrementa en 1 el valor de la variable «Contador», que lleva la cuenta de las letras que contiene cada línea de texto (7.67).

Tras ejecutar estos subprogramas, al final del bucle se da valor 0 a la variable «Indice».

7.9.4.0.3. Bucle 'Espacio' : Este bucle infinito, espera a que se pulse el pulsador 2, cuando es así, concatena un espacio al final de la cadena «AntCar» e imprime en pantalla, incrementando también la variable «Contador», de manera análoga al funcionamiento del bloque 'Anexar'.

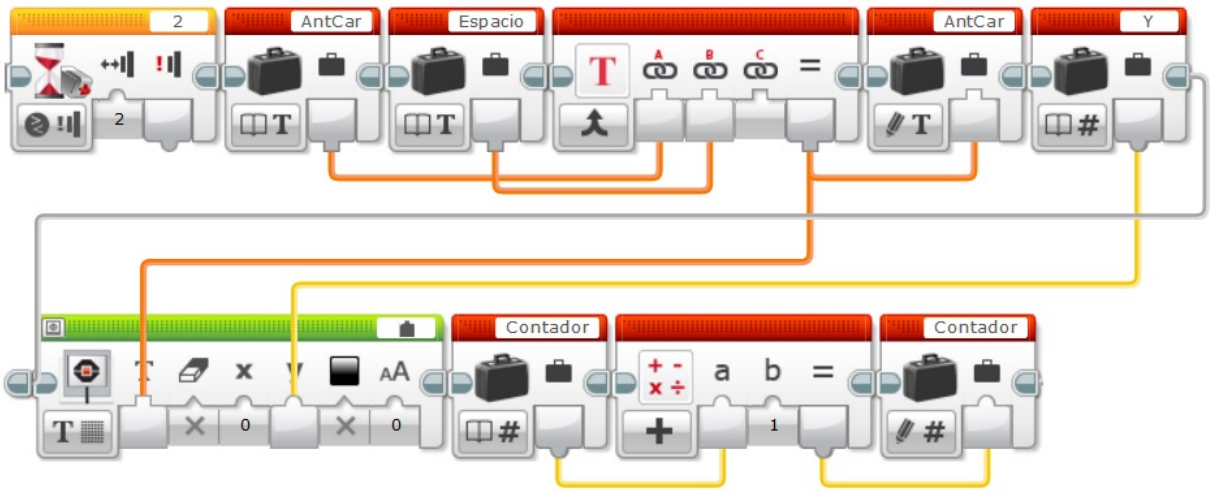


Figura 7.68 – Telégrafo: Bucle 'Espacio'

7.9.4.0.4. Bucle 'RetCarro' : Este bucle controla el salto de línea, éste puede ser automático, cuando el valor de la variable «Contador» alcanza 22, o bien manual, cuando se pulsa el botón central del brick, ambas opciones se evalúan mediante un bloque lógico configurado como OR. Cuando el resultado de esta operación OR es verdadero se ejecuta el bloque «RetCarro.»

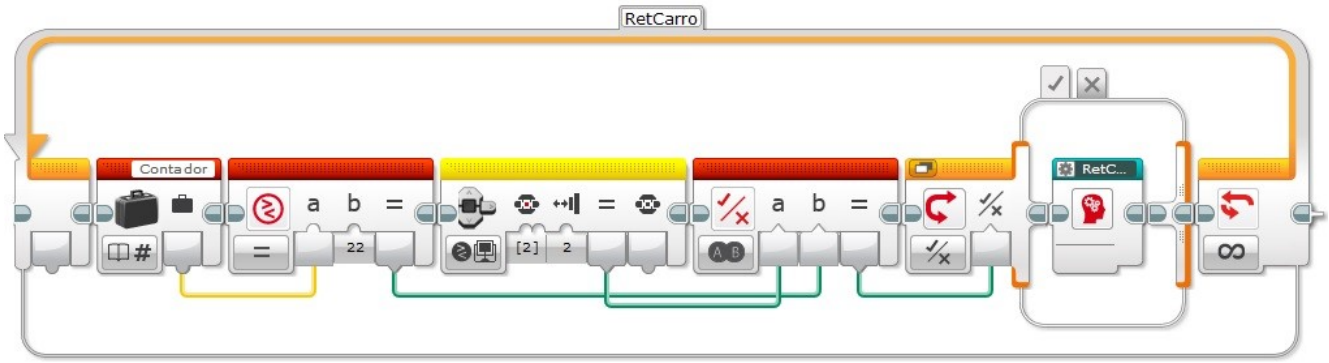


Figura 7.69 – Telégrafo: Bucle 'RetCarro'

- Bloque 'RetCarro':

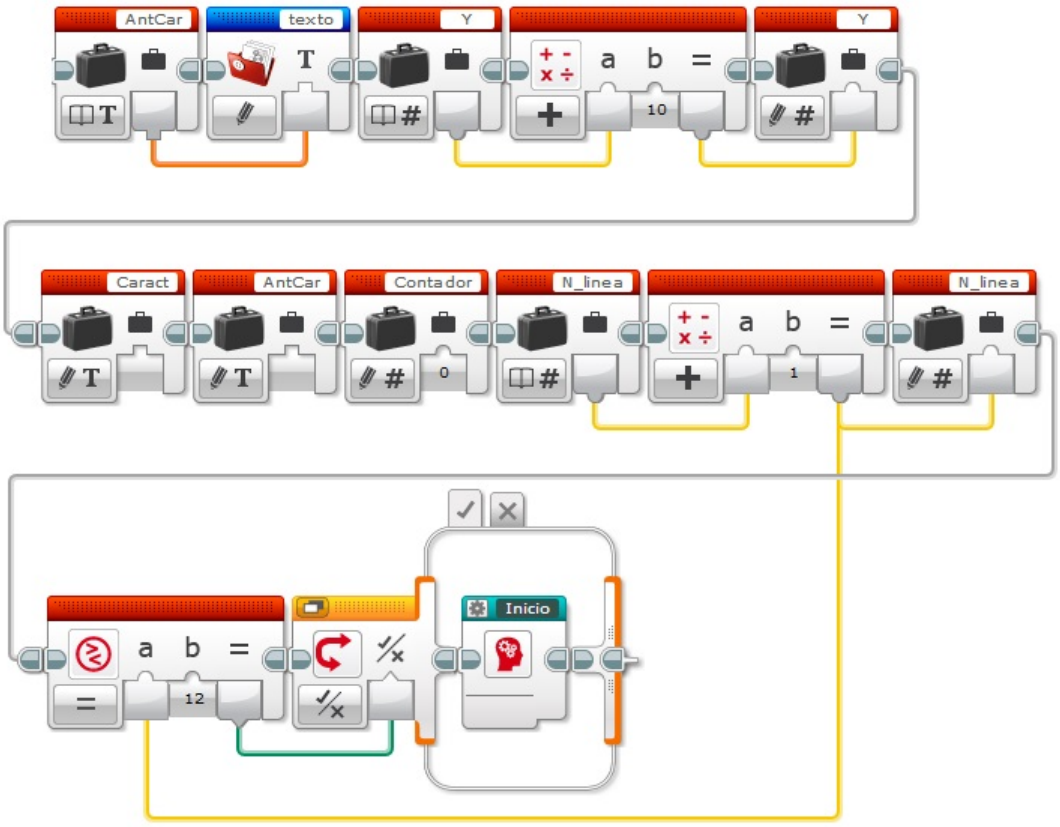


Figura 7.70 – Telégrafo: Bloque 'RetCarro'

Guarda la cadena de caracteres «AntCar» en el archivo texto, incrementa la variable «Y» en 10 unidades, haciendo que la siguiente impresión en pantalla se haga en la siguiente línea.

Pone a 0 la variable «Contador» y borra las variables tipo texto «Caract» y «AntCar».

Incrementa el número de línea («N_linea»), cuando esta llega a 12, ejecuta el bloque 'Inicio', explicado anteriormente (7.70).

7.9.4.0.5. Bucle 'Borrar Todo' : Este bucle infinito espera a que se pulse el botón (4) del brick, cuando esto sucede, borra el archivo texto y ejecuta el bloque 'Inicio'.

Figura 7.71 – Telégrafo: Bucle 'Borrar Todo'

7.9.4.0.6. Bucle 'Borrar ultima' : Este bucle espera a que se pulse el botón (1) del brick, una vez pulsado, imprime en pantalla la cadena de caracteres «Anterior» y sobrescribe el texto «AntCar» con el texto «Anterior». Por último, decrementa en 1 el valor del contador.

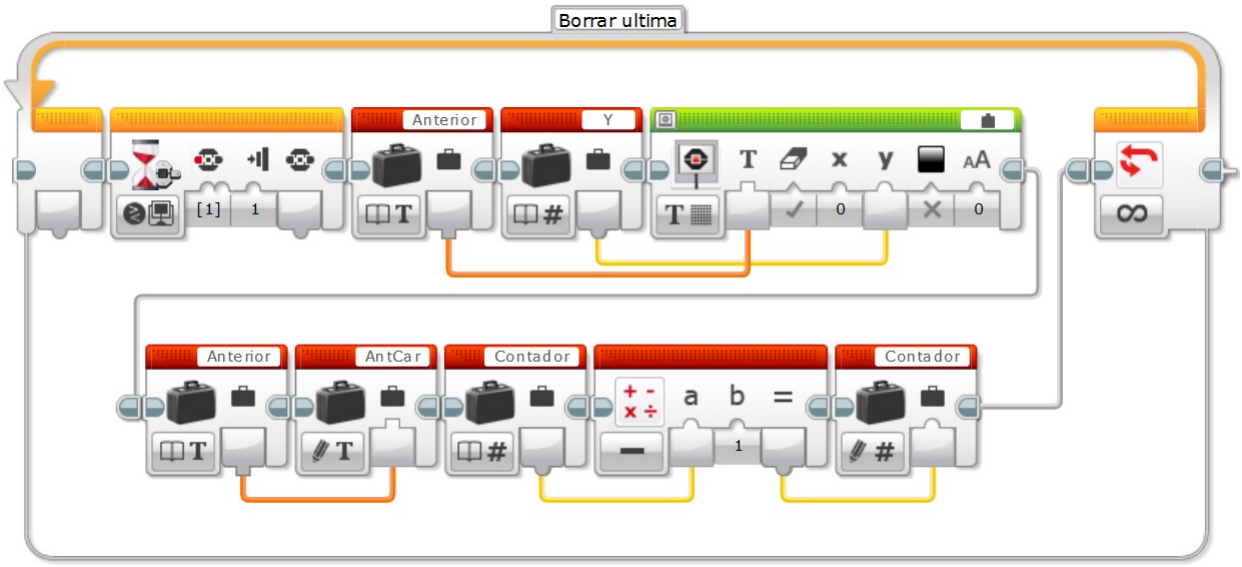


Figura 7.72 – Telégrafo: Bucle 'Borrar ultima'

7.10. Telégrafo/Impresora

7.10.1. Introducción

Ampliaremos el programa anterior para que, además de las funciones del ejemplo anterior, gobierne un robot que escriba el texto en papel.

El robot estará formado por una estructura a modo de cabezal donde colocaremos un rotulador o bolígrafo. Este utensilio está conectado a una cremallera activada por un motor mediano que lo desplaza hacia arriba y abajo.

El cabezal está unido a una cinta gobernada por un motor grande, encargado de realizar el movimiento en el eje X. Para el movimiento en el eje Y, un motor grande mueve un eje con cuatro ruedas que presionan el papel hacia abajo, haciendo q se deslice sobre la superficie donde está apoyado.

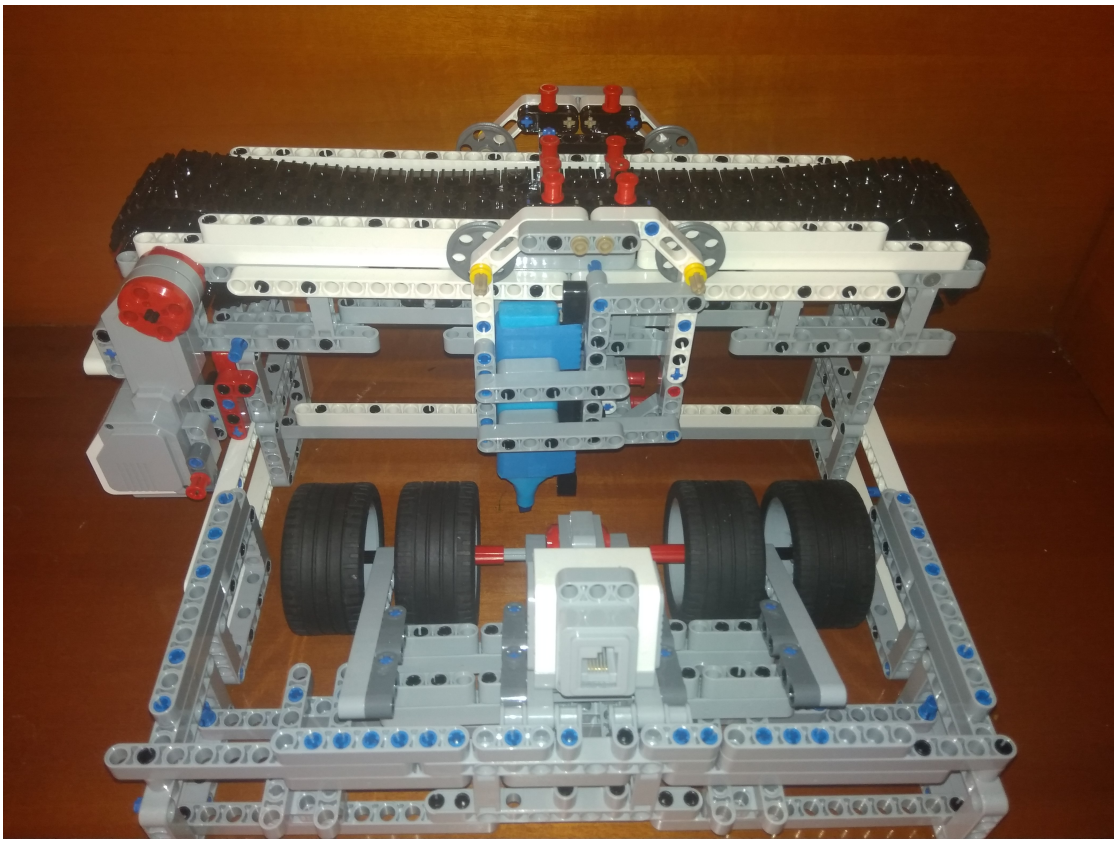


Figura 7.73 – Robot 'impresora'

7.10.2. Sensores y actuadores

Además de los dos pulsadores utilizados del mismo modo que en el ejemplo anterior, utilizaremos como actuadores:

- Un motor grande (Puerto C) para desplazar la cinta que mueve el carro.
- Un motor grande (Puerto D) para desplazar el papel arriba y abajo.

- Un motor mediano (Puerto B) para subir y bajar el utensilio de escritura.

7.10.3. Variables

Nombre	Tipo	Descripción
Indice	Num.	Indica el número de elementos del vector «Letra».
LeePulso	Bool	Indica cuando se ha acabado de leer una letra en morse.
Puntoraya	Num.	0=punto, 1=raya.
Letra	Vec. Num.	Contiene la letra en código morse.
Caract	Texto	Contiene la letra en formato texto.
AntCar	Texto	Contiene la concatenación de letras en formato texto.
Espacio	Texto	Espacio en blanco.
Contador	Num	Cuenta el número de caracteres por línea de texto.
Y	Num	Coordenada Y para impresión en pantalla.
N_linea	Num	Número de línea.
Anterior	Texto	Guarda la concatenación de texto anterior.
Caracter	Num.	Número que corresponde a cada letra.
VectNum	Vec. Num	Concatenación de la variable numérica «Caracter».
ContImpresion	Num.	Contador de impresión.
VectNumIni	Vec. Num.	Valor inicial de los arrays numéricos.
VectNumImp	Vec. Num	Copia del array «VectNum» para impresión.
Grados	Num	Medida en grados del desplazamiento de la cinta.

Tabla 7.8 – Telégrafo/Impresora: Variables

7.10.4. Descripción del programa

La estructura del programa es idéntica a la del ejemplo anterior, por lo que comentaremos tan sólo los bloques de código añadidos a mayores sobre el ejemplo anterior, necesario para gobernar el robot 'impresora'.

El algoritmo para decodificar el código Morse es el mismo pero además de adjudicar una letra a una variable tipo texto, adjudica también un valor a una variable numérica.

En función del valor de esta variable el robot ejecutará un subprograma con los movimientos coordinados necesarios para dibujar cada letra, siendo las letras de la A a la Z los números del 1 al 26. El espacio se corresponde con el valor numérico 28. El valor 0 no realiza ninguna acción.

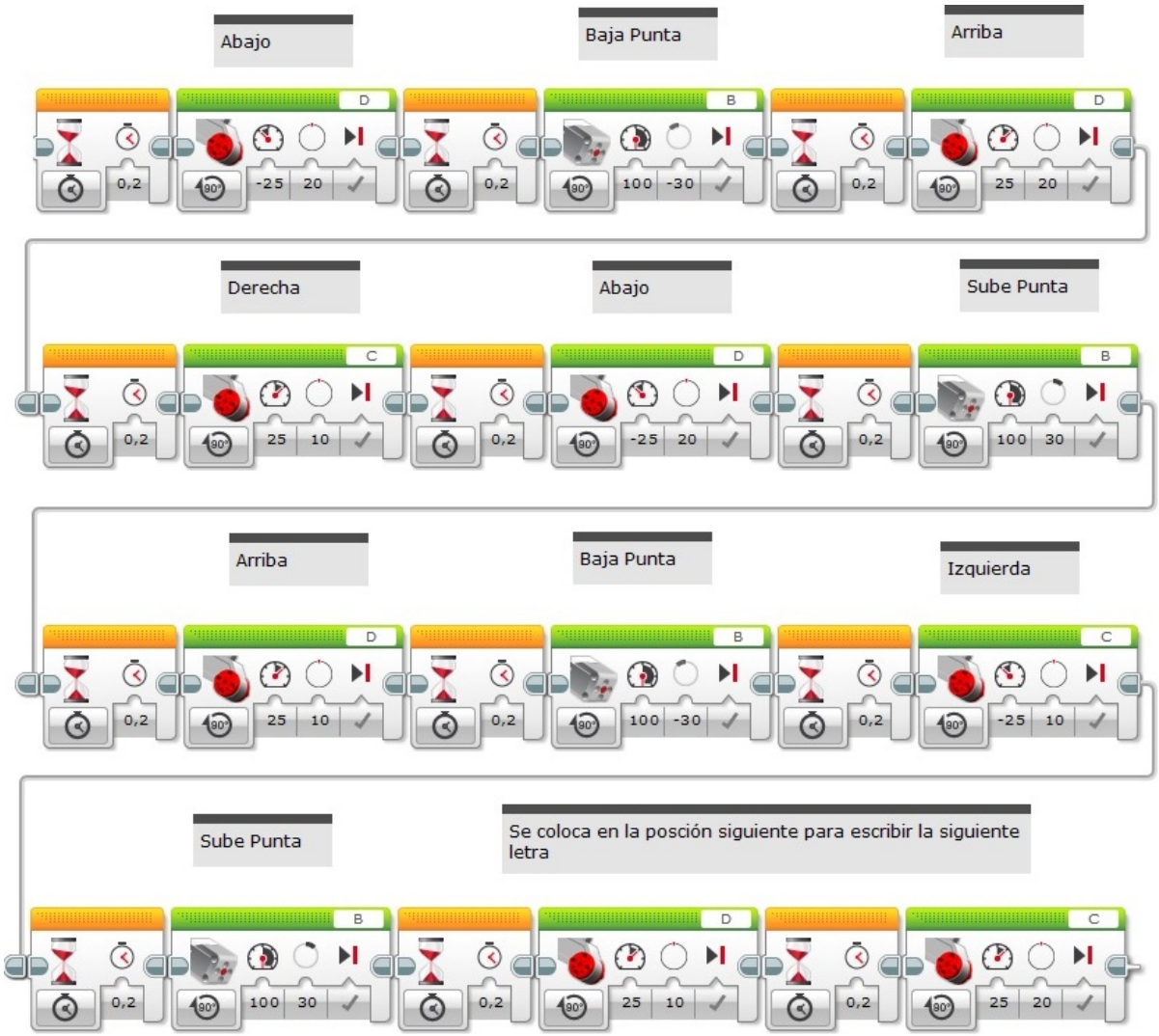


Figura 7.74 – Telégrafo/Impresora: Subprograma 'A'

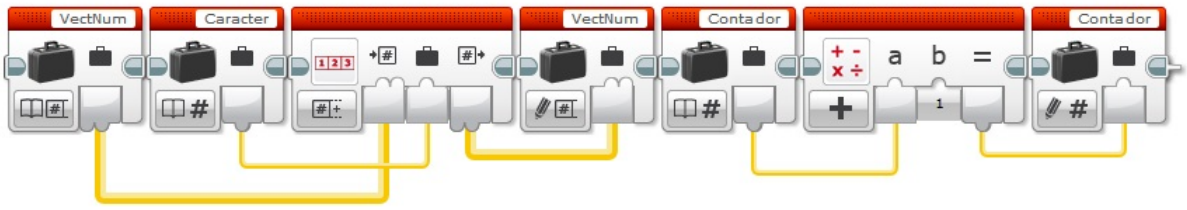


Figura 7.75 – Telégrafo/Impresora: Bloque 'Anexar'

En el bloque 'Anexar', además de concatenar las cadenas de caracteres al igual que en el ejemplo anterior, utiliza el bloque de operaciones secuenciales para anexar los valores numéricos en el array «VectNum» (7.75).

En el bloque 'RetCarro', vuelca el contenido de «Contador» en «ContImpresion», de este modo podemos reiniciar la variable «Contador». Esto sirve para que mientras el robot escribe una línea de texto, podamos introducir la línea siguiente.

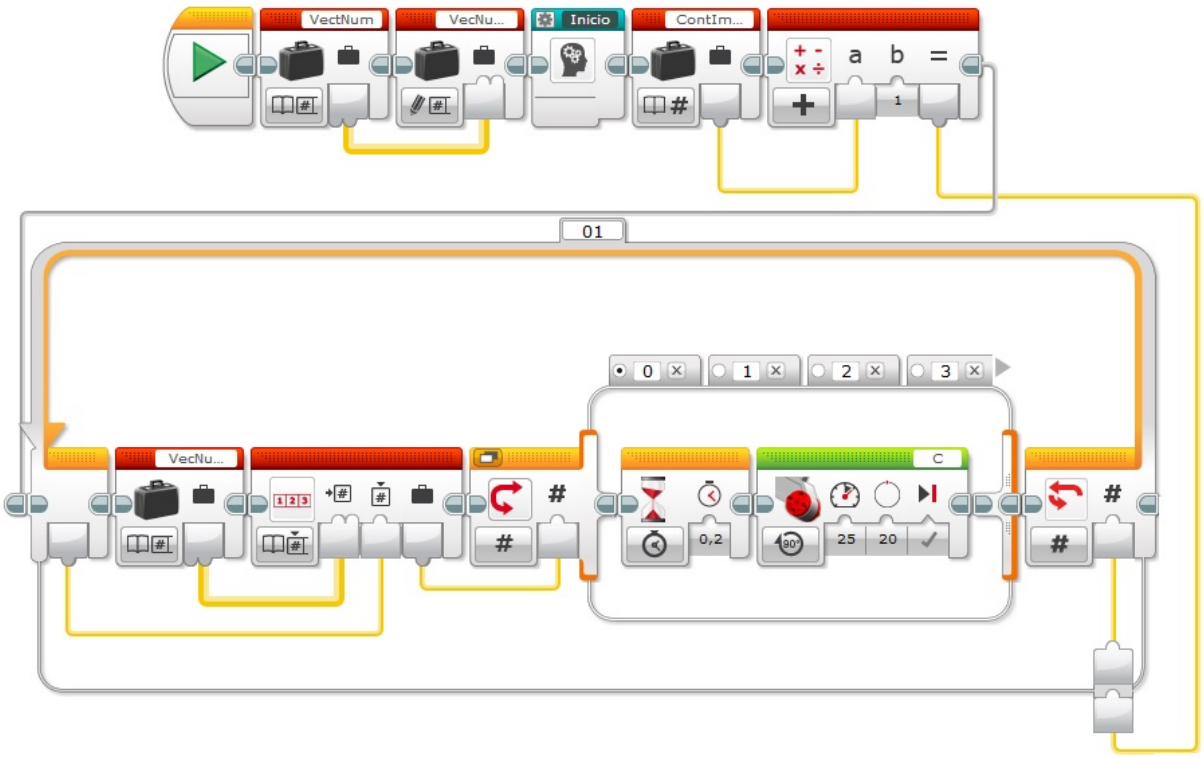


Figura 7.76 – Telégrafo/Impresora: Bloque 'Imprime'(Detalle)

Tras el bloque 'RetCarro' añadimos un bloque llamado 'Imprime' que gobierna el robot. Este bloque comienza volcando el contenido del vector numérico «VectNum» en «VectNumImp», trabajando con éste, ya que «VectNum» se sobreescribe a medida que se introduce texto.

Tras ello, ejecuta el bloque 'Inicio', que reinicia variables y borra cadenas de caracteres y además toma una lectura del encoder del motor C, encargado de mover la cinta, guardándola en la variable «Grados».

Acto seguido entrará en un bucle que se ejecutará tantas veces como caracteres contenga la línea de texto, evaluará las posiciones del vector «VecNumImp» una a una y en función del valor numérico ejecutará el subprograma correspondiente a cada letra (o espacio), 7.76.

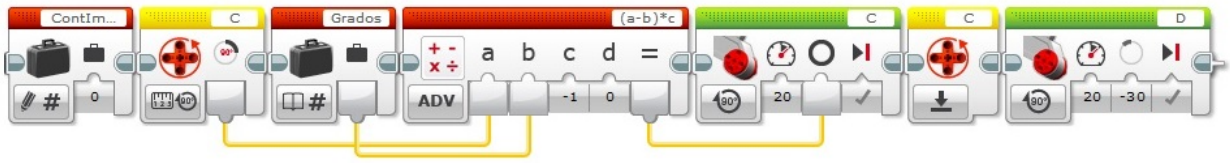


Figura 7.77 – Telégrafo/Impresora: Bloque 'Imprime'(Detalle)

Una vez sale del bucle, reinicia el contador de impresión y en la siguiente línea, toma la lectura del motor C almacenada en la variable «Grados» y la resta a la lectura actual de dicho motor. De esta manera obtenemos el número de grados que se ha desplazado la cinta, multiplicando este valor por -1 para invertir el sentido, hacemos que la cinta retroceda el mismo

número de grados. Por último activamos el motor D, para desplazar el papel hacia arriba, terminando así el salto de línea (7.77).

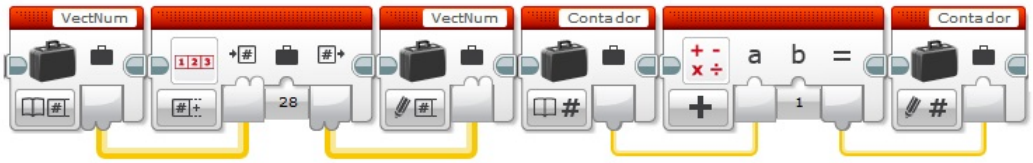


Figura 7.78 – Telégrafo/Impresora: Bucle 'Espacio'(Detalle)

En el bucle 'Espacio', utiliza un bloque de operaciones secuenciales, para anexar el valor 28 (que corresponde a espacio) al final del array «VecNum», tras ello incrementa el contador en una unidad (7.78).

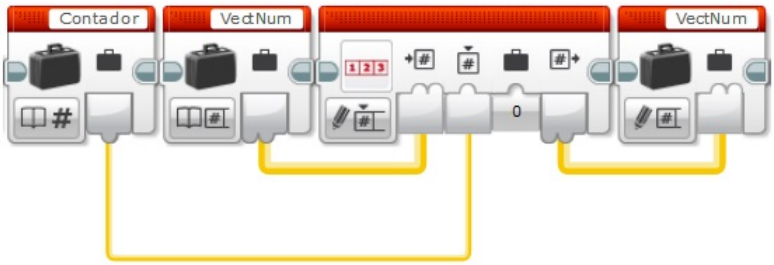


Figura 7.79 – Telégrafo/Impresora: Bucle 'Borrar ultima'(Detalle)

En el bucle 'Borrar ultima', para borrar la última posición del array numérico, toma el valor de «Contador» y sobrescribe dicha posición del array «VectNum» con el valor 0, que en el subprograma 'Imprime' no realiza ninguna acción (7.79).

8 RESULTADOS FINALES

Dadas las características particulares del presente TFG, la naturaleza gráfica del lenguaje de programación EV3-G y la extensión de los programas, éstos se presentarán completos en la sección Planos para su mejor visualización y comprensión, utilizando el documento Memoria para ofrecer una explicación pormenorizada de los mismos.

En el CD adjunto se incluyen, además de los archivos ejecutables y modelos 3D de los robots, el software necesario para la ejecución de los programas y para la visualización de las instrucciones de montaje, así como videos demostrativos para cada robot.

Este contenido estará también disponible en el siguiente enlace:

https://drive.google.com/drive/folders/1f06GP-_LkM2VLjS2sqR8qcN435fxjsvM?usp=sharing

9 ORDEN DE PRIORIDAD ENTRE LOS DOCUMENTOS

Frente a posibles discrepancias, el orden de prioridad de los documentos del TFG, debe ser el siguiente:

1. Planos
2. Memoria
3. Pliego de Condiciones
4. Presupuesto

TÍTULO: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**

ANEXOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: FEBRERO DE 2020

AUTOR: EL ALUMNO

Fdo.: DAVID GÓMEZ OCAMPO

Índice del documento ANEXOS

10 DOCUMENTACIÓN DE PARTIDA

10.1. ASIGNACIÓN DE TRABAJO FIN DE GRADO

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Gómez Ocampo,David*
APELIDOS E NOME:

DNI: [redacted] **Fecha de Solicitud:** Feb2019
DNI: *Fecha de Solicitude:*

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:Librería de software didáctico para un robot Lego

Número TFG: *770G01A164*

TUTOR:(Titor) *Leira Rejas,Alberto Jose*

COTUTOR/CODIRECTOR:

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento*:*

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Miercoles, 18 de Septiembre del 2019

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Gómez Ocampo,David

DESCRIPCIÓN Y OBJETIVO:Se pretende desarrollar una librería de al menos diez aplicaciones para un robot móvil Lego, en grado creciente de complejidad y con sus tutoriales respectivos para aplicación posterior en la enseñanza o en demostraciones prácticas.
El alumno/a tendrá ocasión de implementar y comprobar físicamente su proyecto sobre robots reales.

11 CÁLCULOS

11.1. Cálculo de los parámetros del controlador PID

Para afinar los parámetros del controlador PID, seguiremos los siguientes pasos:

1. Ajustar Ki y Kd a 0, con lo que tendríamos un controlador proporcional simple.

2. Hacer funcionar el robot, si no es capaz de seguir la línea y se desvía de ésta (por defecto), incrementar el valor de Kp. Si oscila demasiado hasta el punto de no ser capaz de seguir la línea (por exceso), decrementar Kp.
   Cuando obtengamos un valor de Kp que haga que el robot siga la línea, con oscilaciones, pero sin perder la trayectoria, consideraremos este valor de Kp como valor de ganancia crítico (Kc).

3. Usando este valor Kc como ganancia proporcional Kp, trataremos de determinar el tiempo de oscilación (Pc), este tiempo es que tardará el robot en oscilar de un lado de la línea a otro y de nuevo a la posición inicial, podemos cronometrarlo o usar el bloque de recogida de datos y visualizar la lectura del sensor óptico frente al SetPoint y el tiempo para ser más precisos.

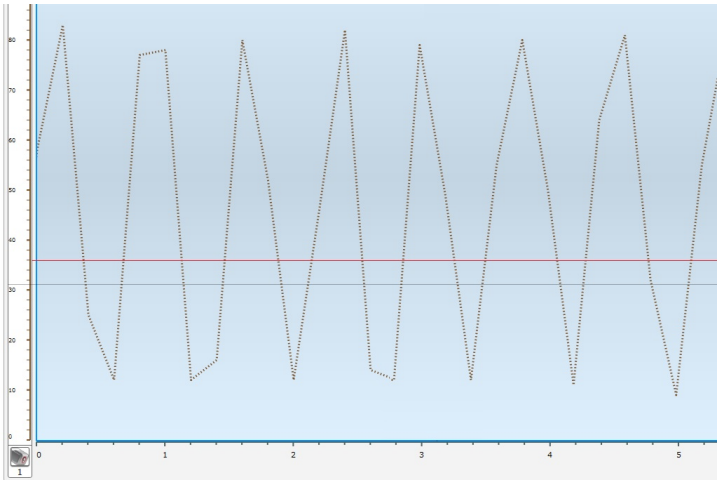


Figura 11.1 – Medida de Pc

4. Con estos datos de Kc y Pc, podemos consultar la tabla de valores de ganancias Ziegler-Nichols y obtener unos valores para Kp, Ki y Kd.

Para el cálculo de los parámetros se necesitaría tener en cuenta el diferencial de tiempo, por ejemplo:

$$integral = integral + error \times (dT) \tag{11.1}$$

En el algoritmo que hemos implementado no tenemos en cuenta el término «dT», que sería el tiempo transcurrido entre iteraciones, esto es debido a que estamos considerando el tiempo de iteración de bucle como constante (aunque puede variar, esta variación es pequeña), por lo que teniendo en cuenta este tiempo a la hora de calcular los parámetros Kp, Ki y Kd, podríamos considerar:

$$integral = integral + error \tag{11.2}$$

Para hallar el tiempo promedio prodremos hacer un pequeño programa a modo de prueba que ejecute un bucle un número determinado de veces y devuelva un valor medio.

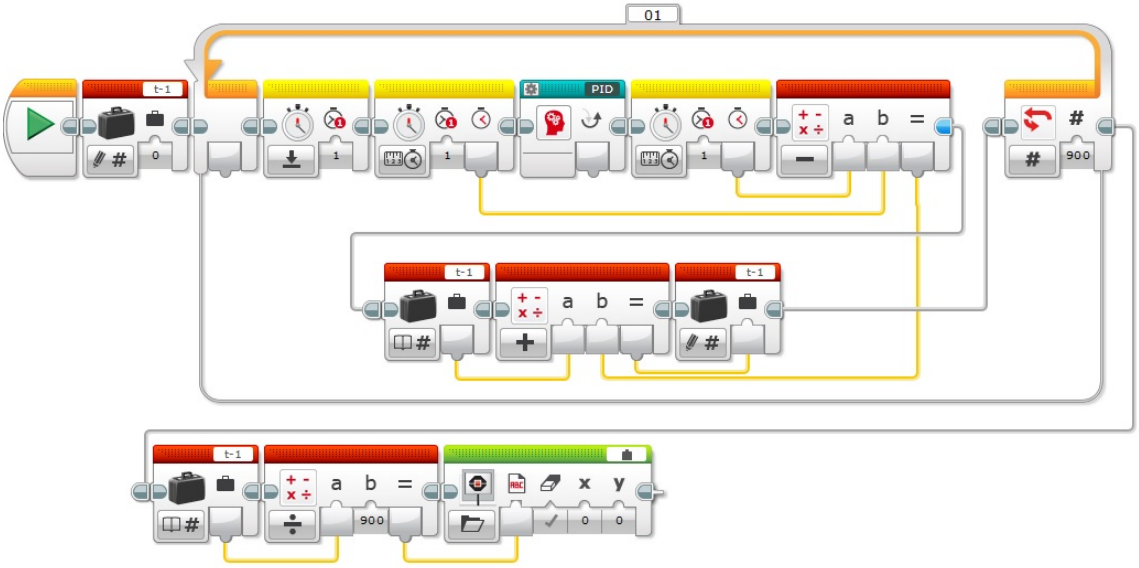


Figura 11.2 – Obtención del tiempo promedio de iteración

Una vez conocemos Kc, Pc y dT iremos a la tabla y realizaremos los cálculos.

Tipo de Control	Kp	Ki	Kd
P	$0{,}5K_c$	0	0
PI	$0{,}45K_c$	$1{,}2K_pdT/Pc$	0
PD	$0{,}80K_c$	0	$K_pP_c/(8dT)$
PID	$0{,}60K_c$	$2K_pdT/P_c$	$K_pP_c/(8dT)$

Tabla 11.1 – Ziegler-Nichols: Valores de ganancia

$$K_p = 0{,}60K_c = (0{,}60)(1{,}75) = 1{,}05 \tag{11.3}$$

$$K_i = 2K_pdt/P_c = 2(1{,}05)(0{,}009)/(0{,}8) = 0{,}024 \tag{11.4}$$

$$K_d = K_p P_c / 8dT = (1{,}05)(0{,}8)/((8)(0{,}009)) = 11{,}66 \quad (11.5)$$

Tras realizar varias pruebas ajustamos los parámetros a:

K_p=1.15 , K_i=0.05 y K_d=5

Si necesitamos hacer algún reajuste manualmente, lo haremos teniendo en cuenta el siguiente criterio:

Parámetro	t_subida	Sobreoscilación	t_est.	Error en R.P.
K_p	Disminuye	Aumenta	No varía	Disminuye
K_i	Disminuye	Aumenta	Aumenta	Elimina
K_d	No varía	Disminuye	Disminuye	No varía

Tabla 11.2 – Ziegler-Nichols: efectos de incrementar los parámetros

- t_subida: El tiempo de subida, determina cómo de rápido el robot trata de corregir un error, depende en gran medida de Kp, un valor alto de Kp hace que responda rápido, pero puede causar sobreoscilación.

- Sobreoscilación: Se traduciría en este caso como cuánto se aleja de la línea cuando trata de corregir un error, la sobreoscilación puede corregirse aumentando el valor de Kd, aunque Kp y Ki también influyen (de manera contraria a Kd).

- t_est.: El tiempo de establecimiento sería el tiempo que el robot tarda en estabilizarse cuando se encuentra con un cambio en la entrada, en nuestro caso una curva.

- Error en R.P.: El error en régimen permanente, es el error que se mantiene cuando no hay perturbación a la entrada, en nuestro caso, cuando el robot sigue una línea recta, este error, que es causado por la parte proporcional del algoritmo, se elimina al introducir el término integral.

12 ANEXOS EN FUNCIÓN DEL ÁMBITO DE APLICACIÓN DEL TFG

12.1. Seguridad

12.1.1. Consideraciones sobre higiene postural y visual

Tradicionalmente se asocian las lesiones en el ámbito de trabajo con actividades que requieren esfuerzo físico (manejo de cargas, etc. . .), sin embargo, el trabajo prolongado frente a monitores, permaneciendo mucho tiempo sentados, conlleva riesgos para la salud si no se realiza de manera adecuada, por eso conviene seguir unas pautas de higiene postural y cuidado de la vista para evitar el deterioro de nuestra salud.

12.1.1.0.1. Higiene postural : Es el conjunto de normas, cuyo objetivo es mantener la correcta posición del cuerpo, en quietud o en movimiento y así evitar posibles lesiones aprendiendo a proteger principalmente la columna vertebral.

Pautas básicas de higiene postural: A la hora de trabajar con ordenadores, ajustaremos el mobiliario para que nos permita en la medida posible cumplir con las siguientes recomendaciones.

- Cuello: mantener la mirada siempre al frente, la línea de visión debe corresponderse con la parte superior de la pantalla.
- Espalda: mantener su curvatura natural y permanecer apoyados en el respaldo.
- Codos: pegados al cuerpo, manteniendo un ángulo entre 90º y 100º.
- Antebrazos: apoyados sobre el escritorio o apoyabrazos de la silla (si los hubiese).
- Muñecas: relajadas y alineadas con el antebrazo.
- Cadera: manteniendo un ángulo de entre 90º y 100º, con los muslos paralelos al suelo.
- Rodillas: evitar flexionar las piernas, las rodillas deben formar un ángulo mayor a 90º.
- Pies: mantenerlos apoyados en el suelo o en un soporte.

Se recomienda efectuar pausas frecuentes y estiramientos para prevenir la fatiga.

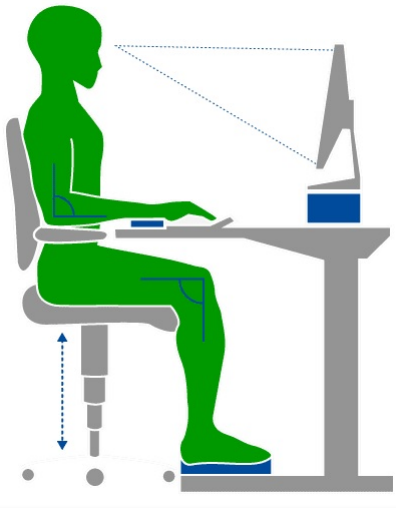


Figura 12.1 – Postura de trabajo correcta

12.1.1.0.2. Higiene visual : La higiene visual consiste en un conjunto de recomendaciones destinadas a controlar los factores que pueden provocar un efecto nocivo sobre la visión. También se conoce como «ergonomía visual». Algunas pautas bautas básicas para el cuidado de la vista son:

- Situar la pantalla lo más alejada posible, siempre que el tamaño de la misma nos permita una correcta visualización y no comprometa nuestra postura.
- Disponer de una buena iluminación ambiental y ajustar el brillo de la pantalla en función del nivel de iluminación disponible.
- Cuando sea posible, ajustar la frecuencia de refresco de la pantalla a una frecuencia alta.
- Utilizar la configuración con la máxima resolución posible.
- Cuando sea posible ajustar el color de fondo a colores menos brillantes (por ej. gris en lugar de blanco).
- Escoger una fuente legible, tanto en tipografía como en tamaño.

Es recomendable mirar de forma regular hacia objetos que estén lejos durante 10-30 s, para relajar el músculo ocular. También se recomienda parpadear, de forma consciente, para hidratar el ojo y/o utilizar lágrimas artificiales o lubricantes oculares.

13 ESTUDIOS CON ENTIDAD PROPIA

Debido a las características particulares del presente TFG, no procede la realización de un Estudio de riesgos laborales o Estudio de Impacto Ambiental.

14 OTROS ANEXOS

En el CD que se adjunta con el presente TFG, se incluyen archivos y documentos que por su naturaleza o por problemas de espacio no pueden ser incluidos en formato papel:

- Archivos ejecutables EV3-G (.ev3).
- Videos demostrativos del funcionamiento de los robots (.mpeg).
- Modelos 3D de los robots (.lxf).
- Software necesario para la visualización de los archivos y generación de los tutoriales:
  - LEGO Mindstorms EV3 v.1.4.2.
  - LEGO Digital Designer v. 4.3.11.

Estos recursos también se encuentran disponibles en la dirección web:
https://drive.google.com/drive/folders/1f06GP-_LkM2VLjS2sqR8qcN435fxjsvM?usp=sharing

TÍTULO: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**

PLANOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

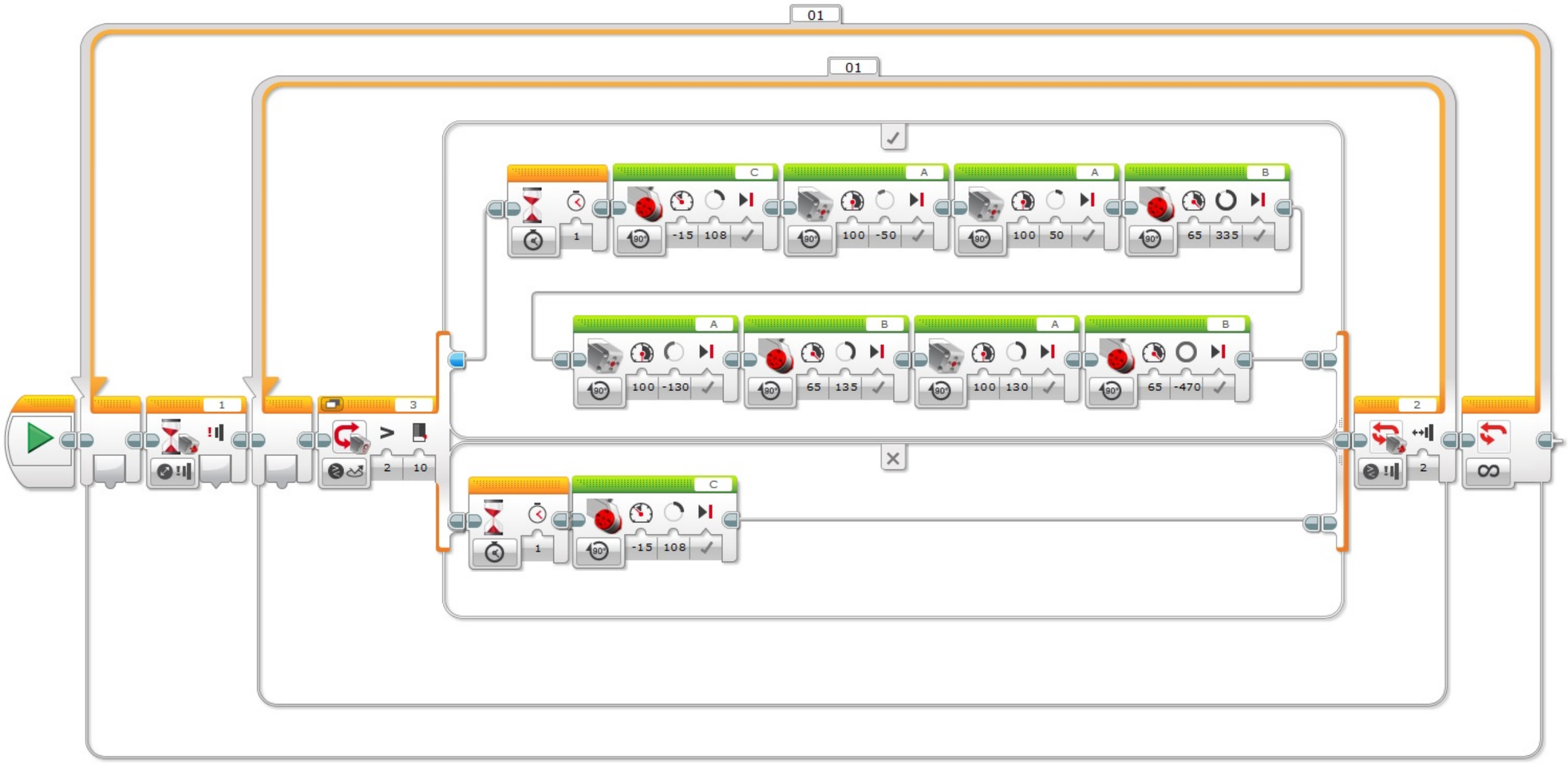
15405 - FERROL

FECHA: FEBRERO DE 2020

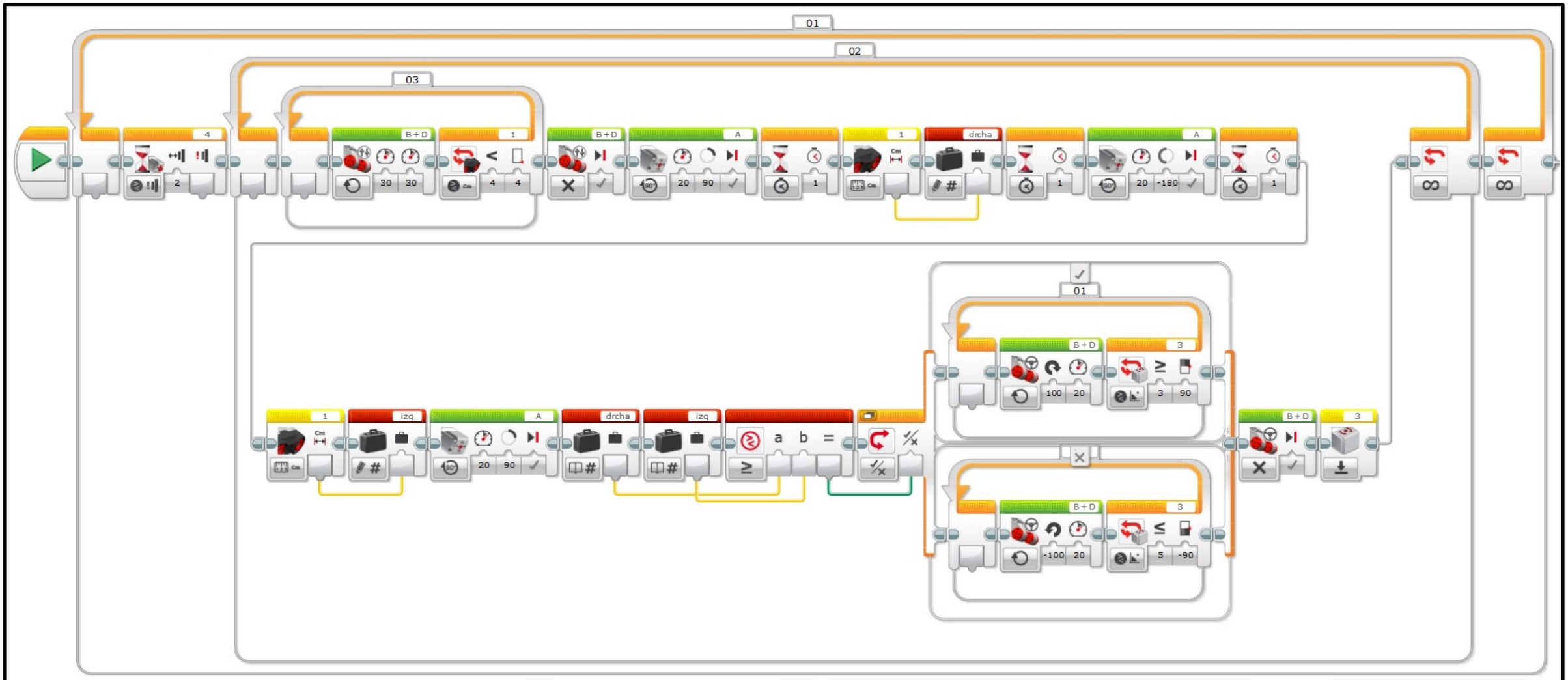
AUTOR: EL ALUMNO

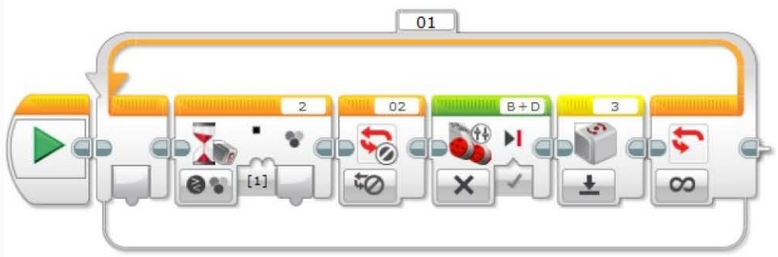
Fdo.: DAVID GÓMEZ OCAMPO

Índice de planos



UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: ROBOT 'PICK & PLACE'	FECHA: FEBRERO 2019
AUTOR: DAVID GÓMEZ OCAMPO — FIRMA:	ESCALA: Sin escala
	PLANO Nº: 01





UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A164

TÍTULO DEL TFG:

LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO

TÍTULO DEL PLANO:

ROBOT ESQUIVA-OBSTACULOS

FECHA: FEBRERO 2019

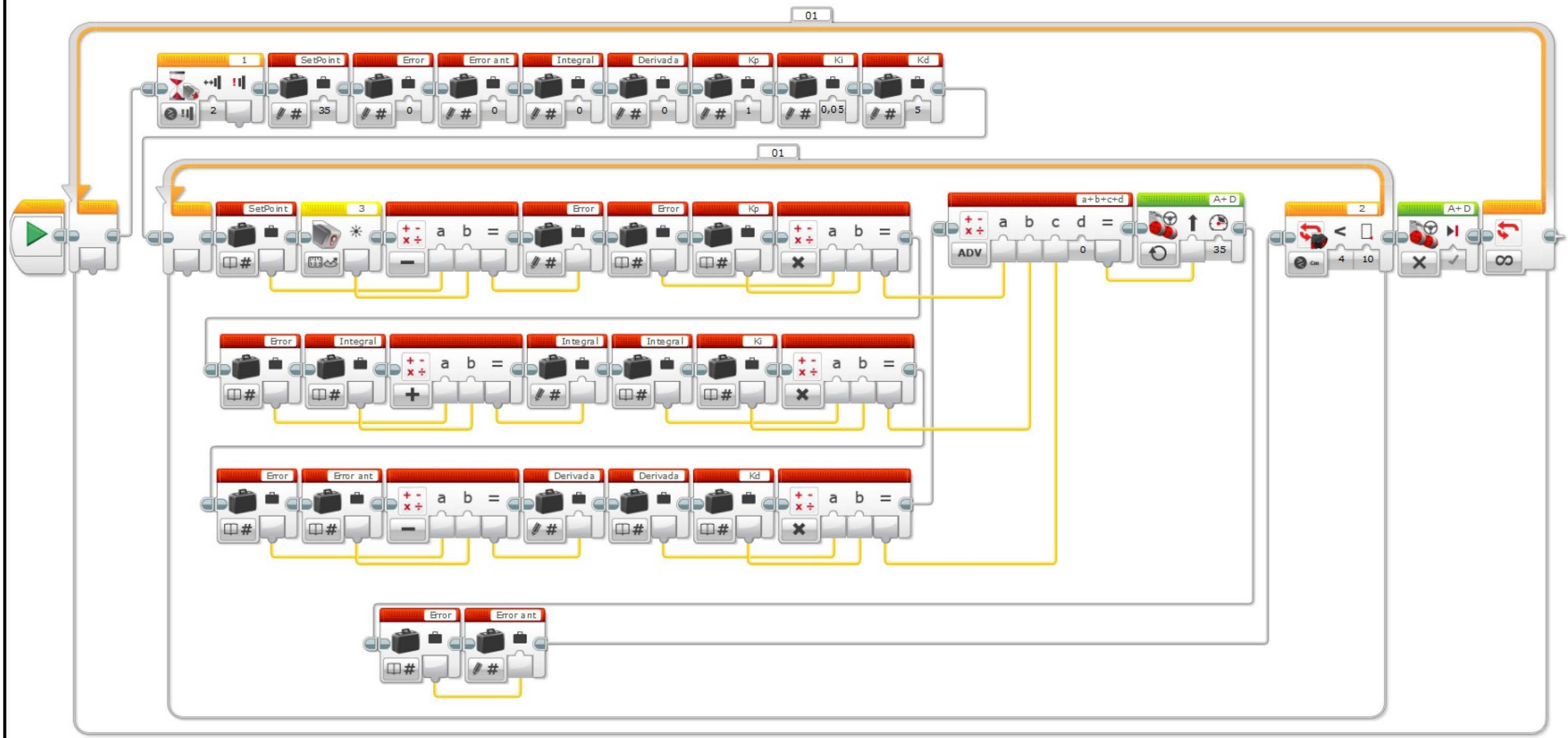
ESCALA: Sin escala

AUTOR:

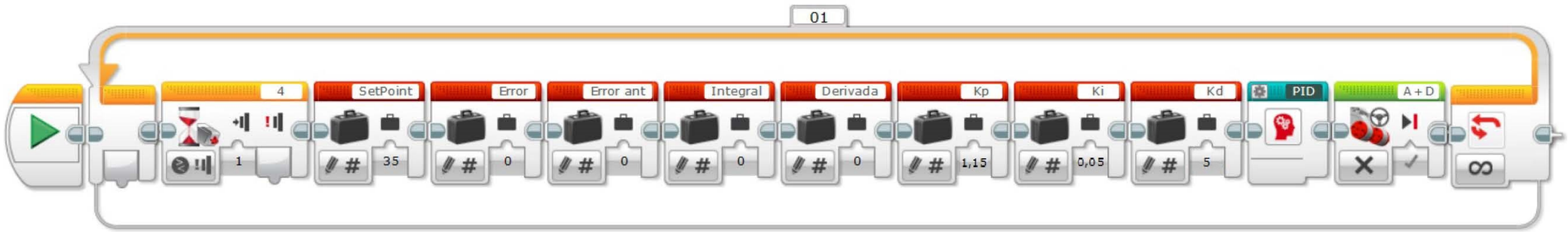
DAVID GÓMEZ OCAMPO

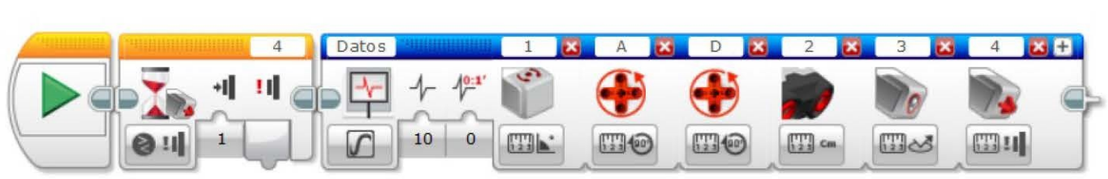
FIRMA:

PLANO Nº: 02



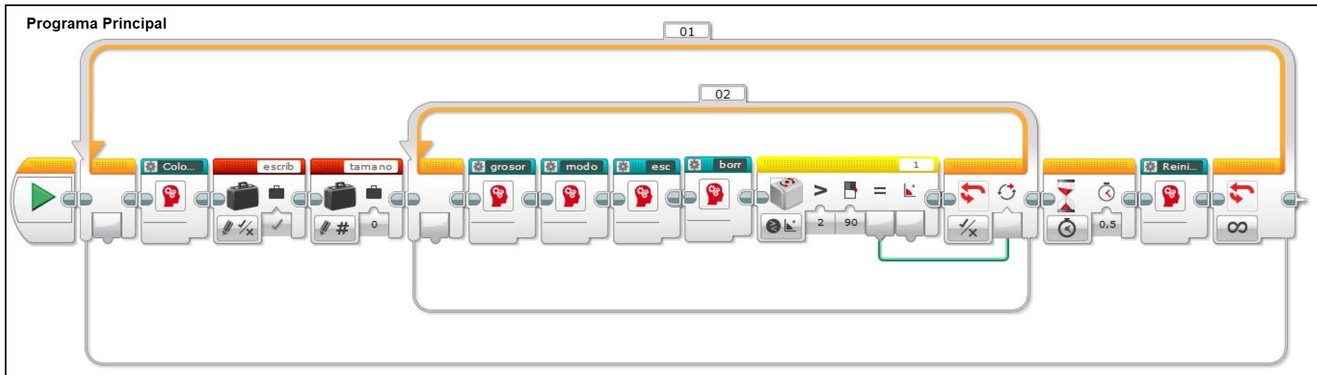
UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: **770G01A164**
TÍTULO DEL TFG: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**	
TÍTULO DEL PLANO: **ROBOT SEGUIDOR DE LÍNEA**	FECHA: FEBRERO 2019 ESCALA: Sin escala
AUTOR: **DAVID GÓMEZ OCAMPO** FIRMA:	PLANO Nº: **03**



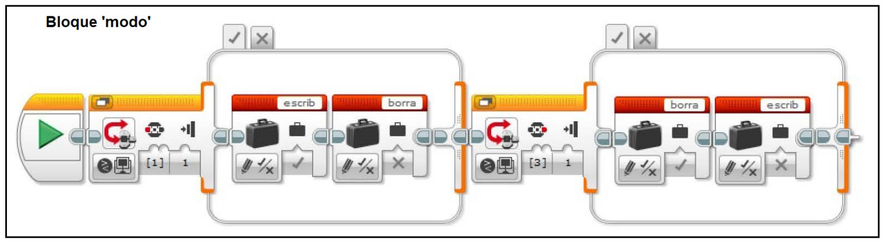


UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: **770G01A164**
TÍTULO DEL TFG: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**	
TÍTULO DEL PLANO: **ROBOT SEGUIDOR DE LÍNEA: RECOGIDA DE DATOS**	FECHA: FEBRERO 2019
	ESCALA: Sin escala
AUTOR: **DAVID GÓMEZ OCAMPO** — FIRMA:	**PLANO Nº: 04**

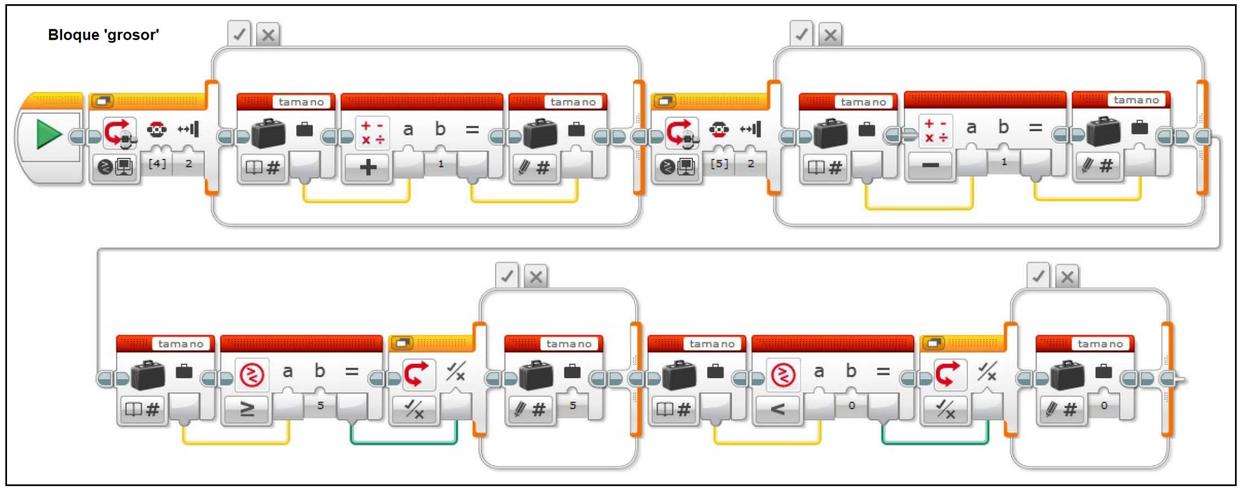
Programa Principal



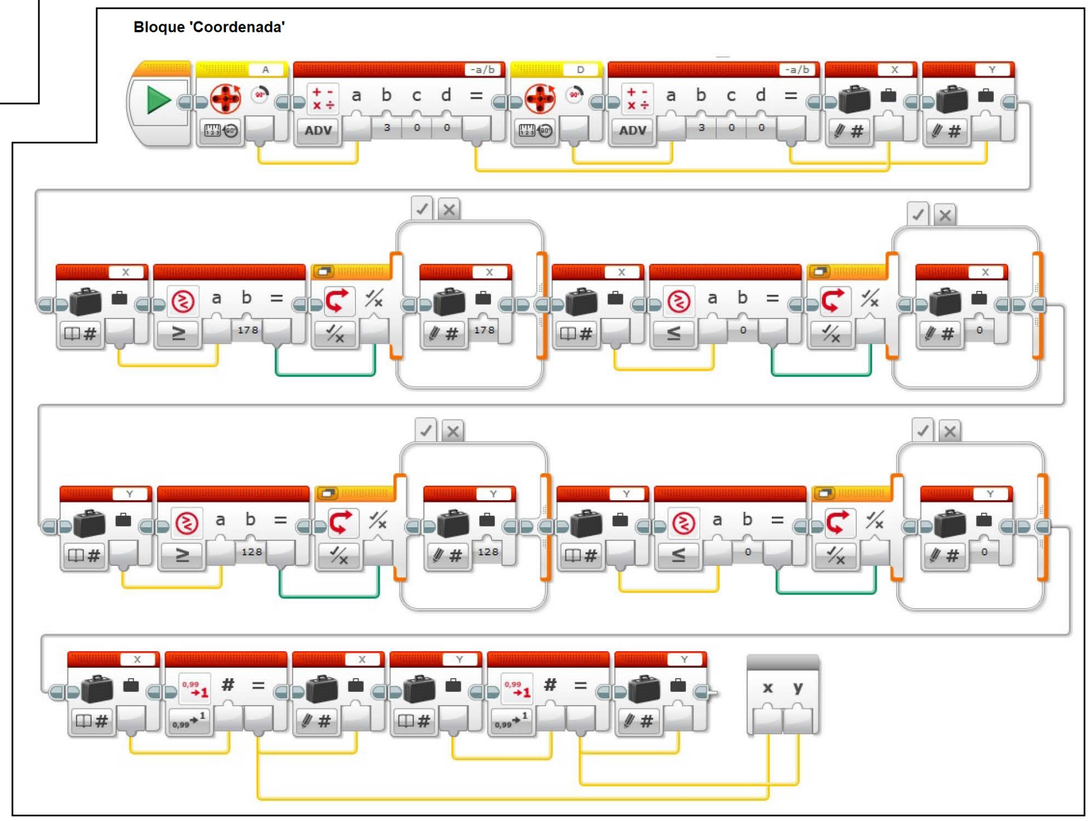
Bloque 'modo'



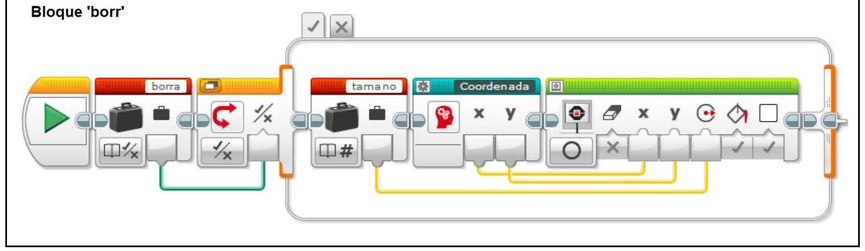
Bloque 'grosor'



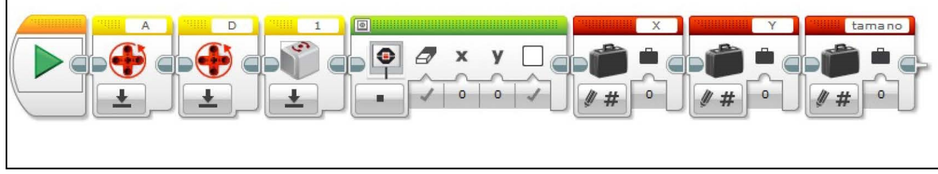
Bloque 'Coordenada'



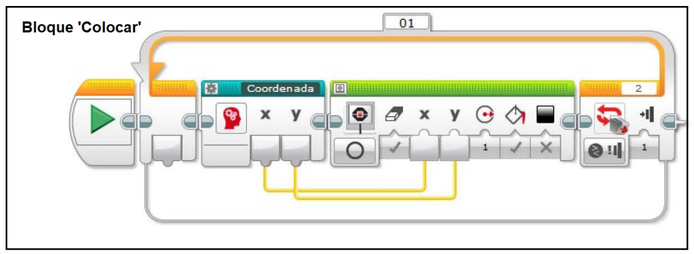
Bloque 'borr'



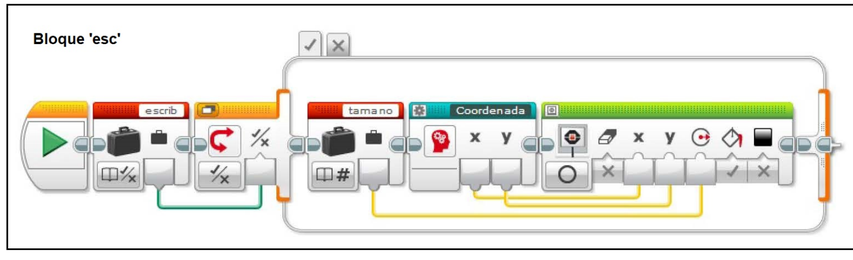
Bloque 'Reiniciar'



Bloque 'Colocar'

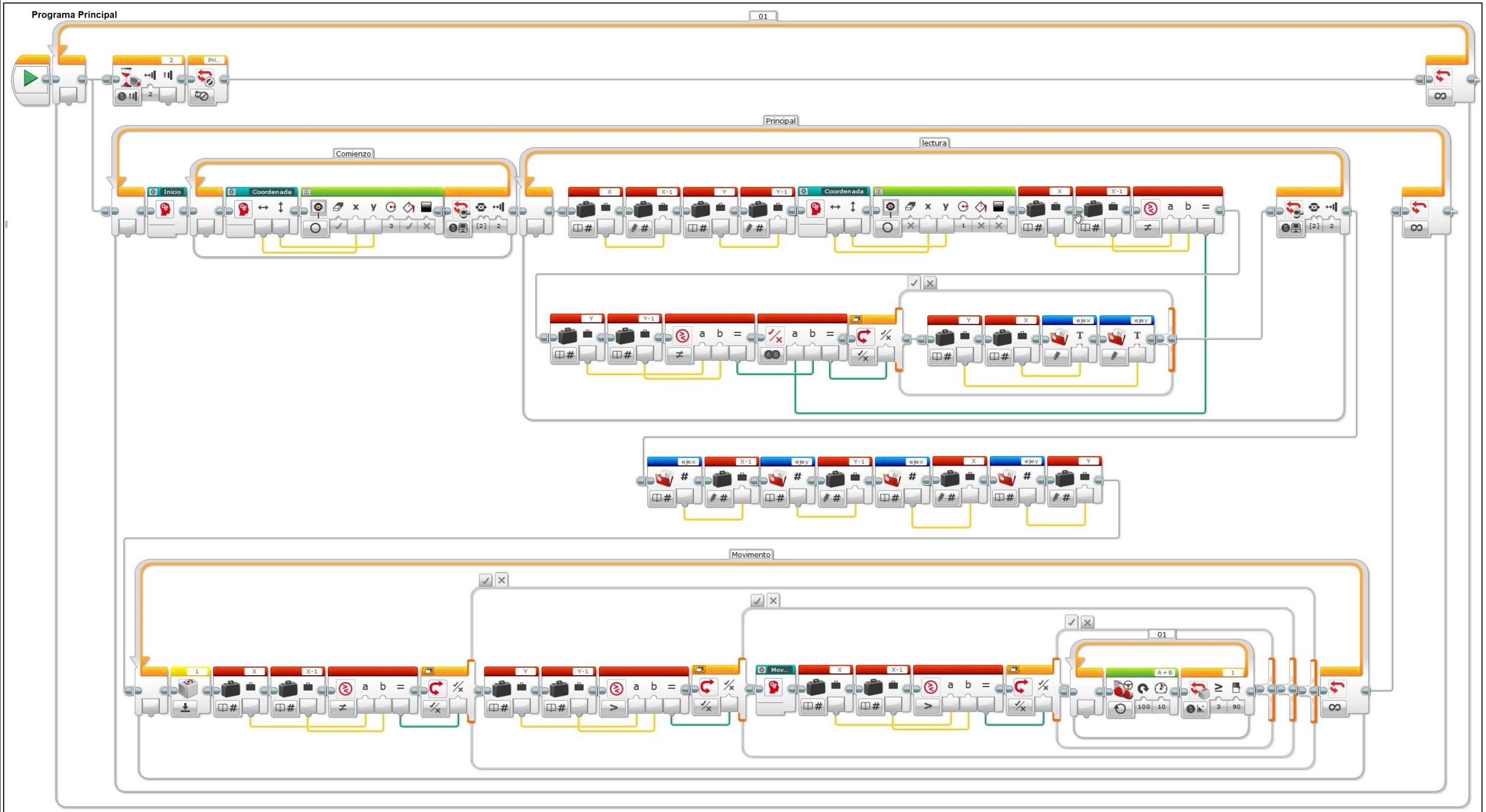


Bloque 'esc'

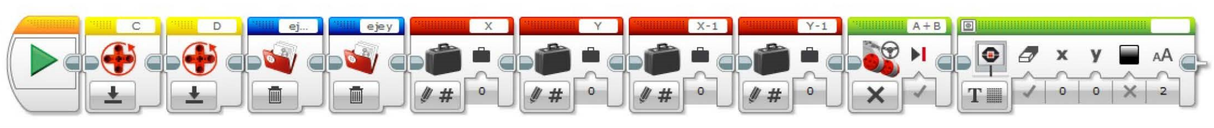


UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: **ROBOT 'TELESKETCH'**	FECHA: FEBRERO 2019 ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO	FIRMA: PLANO Nº: 05

Programa Principal

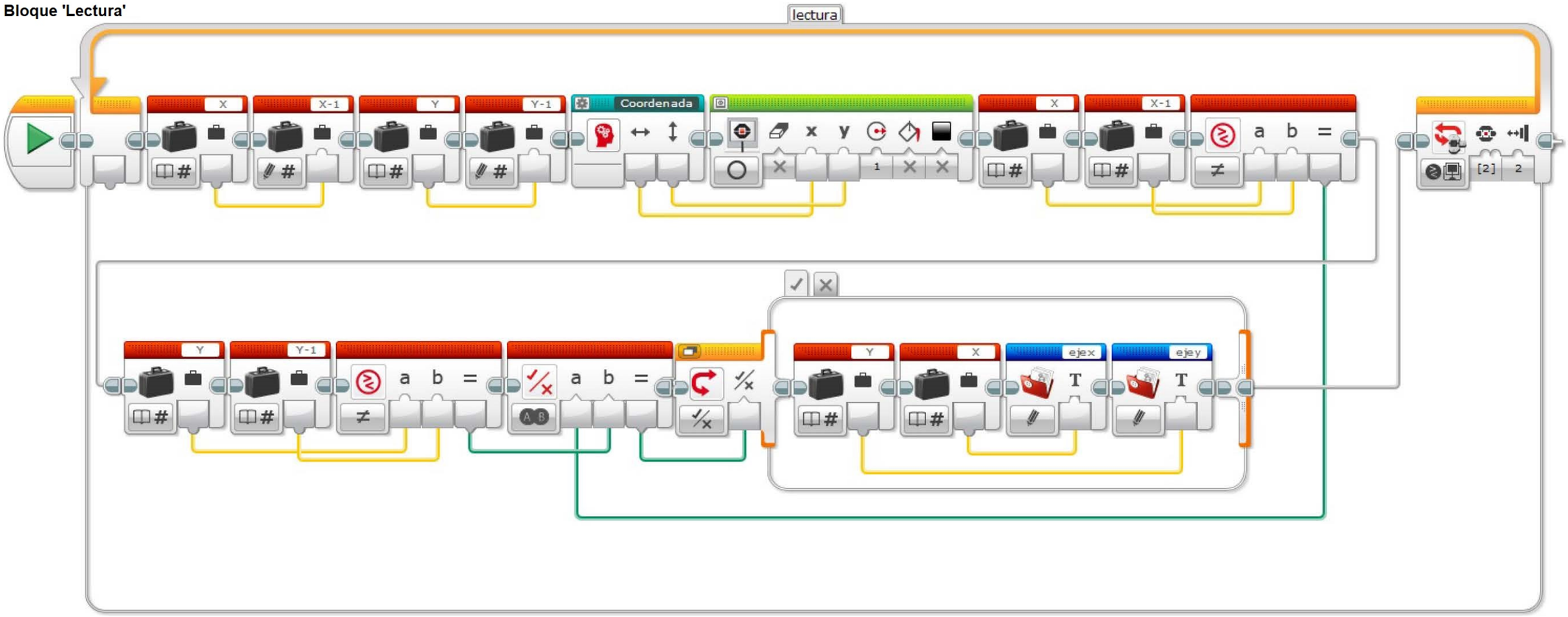


Bloque 'Inicio'

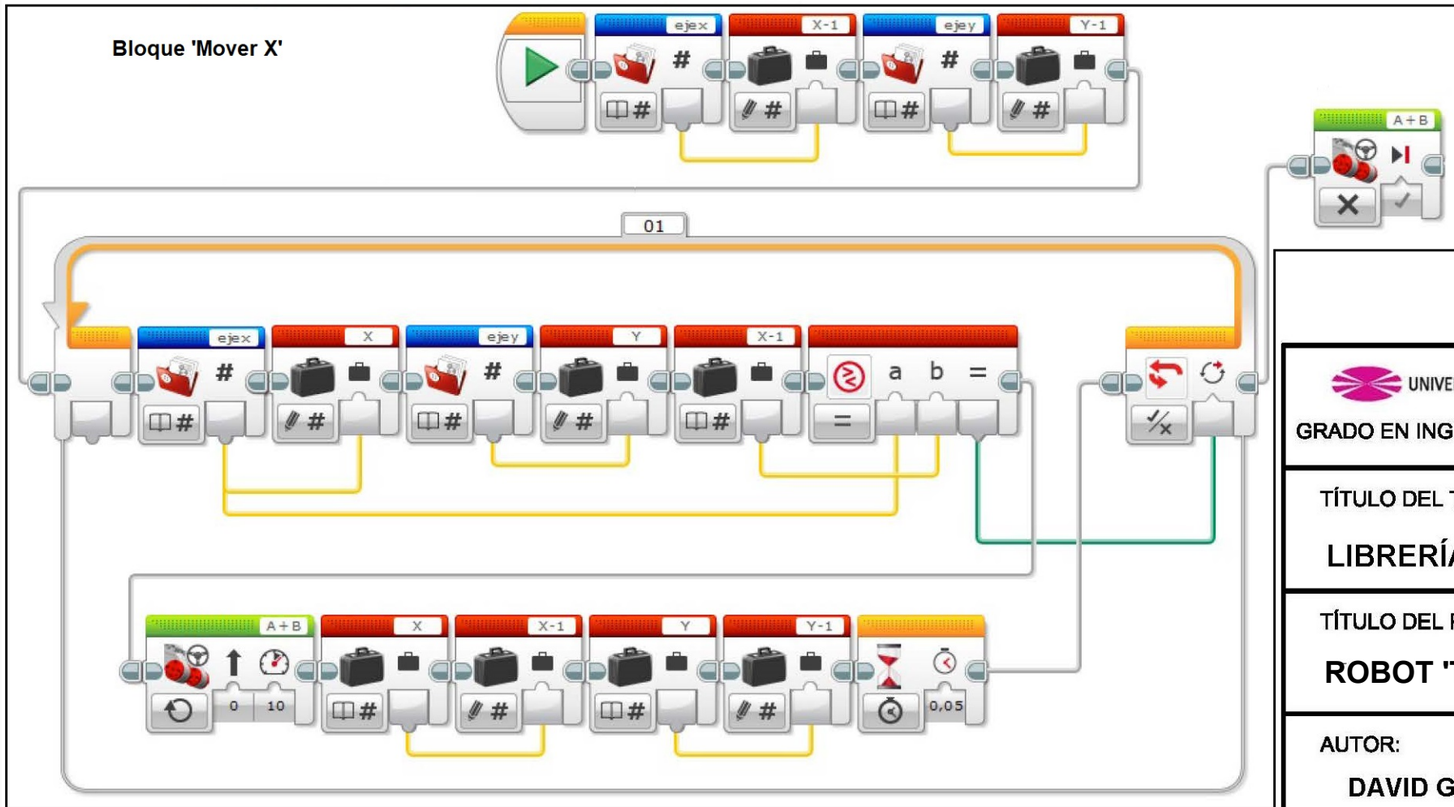


UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: ROBOT 'TRAZADOR' 1	FECHA: FEBRERO 2019 ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO FIRMA:	PLANO Nº: 06

Bloque 'Lectura'



Bloque 'Mover X'



Nota: El bloque 'Mover Y' tiene una estructura idéntica al bloque 'Mover X', compara los valores de "Y" e "Y-1" en lugar de "X" y "X-1".

UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A164

TÍTULO DEL TFG:

LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO

TÍTULO DEL PLANO:

ROBOT 'TRAZADOR' 2

FECHA: FEBRERO 2019

ESCALA: Sin escala

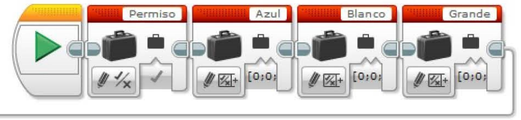
AUTOR:

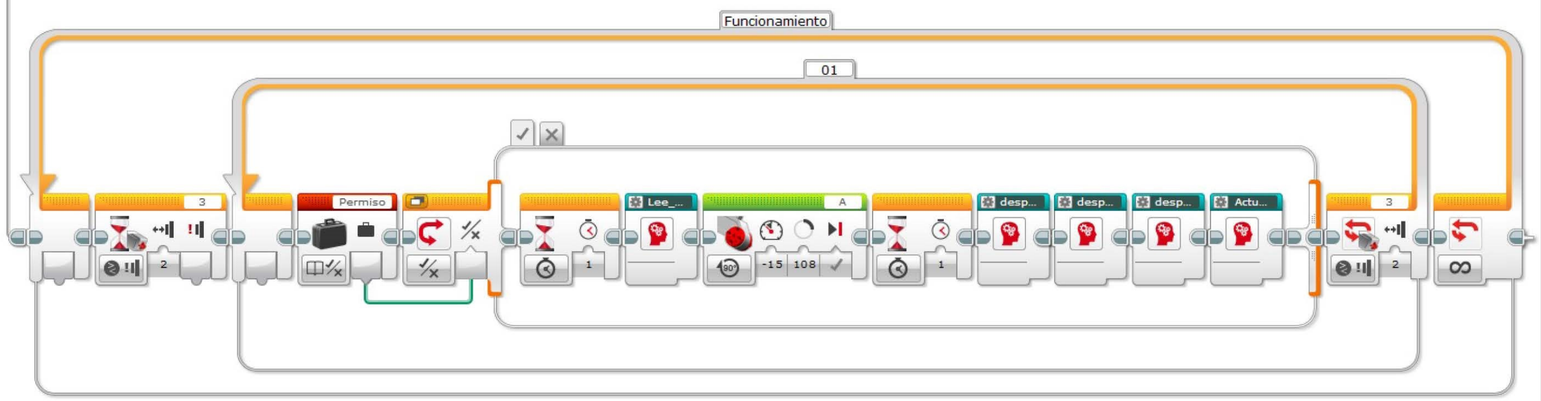
DAVID GÓMEZ OCAMPO

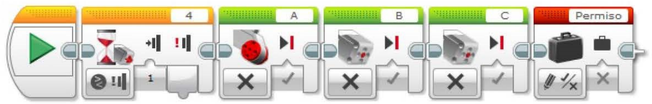
FIRMA:

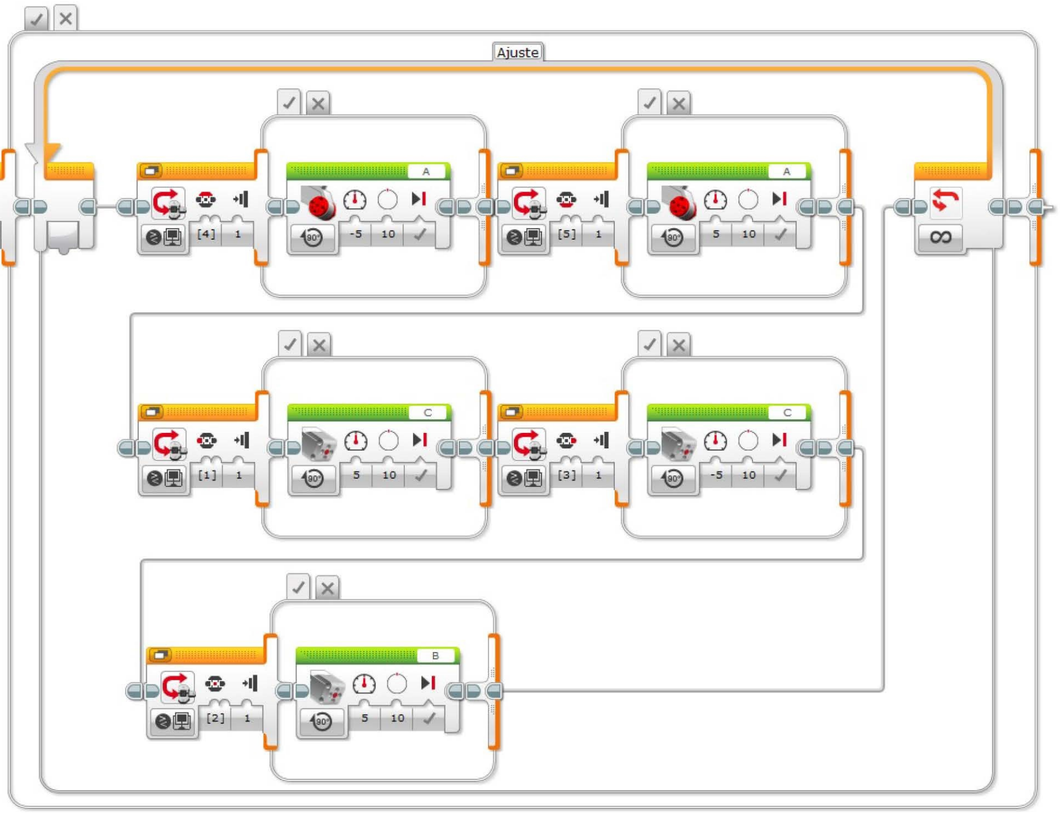
PLANO Nº: 07

Programa Principal

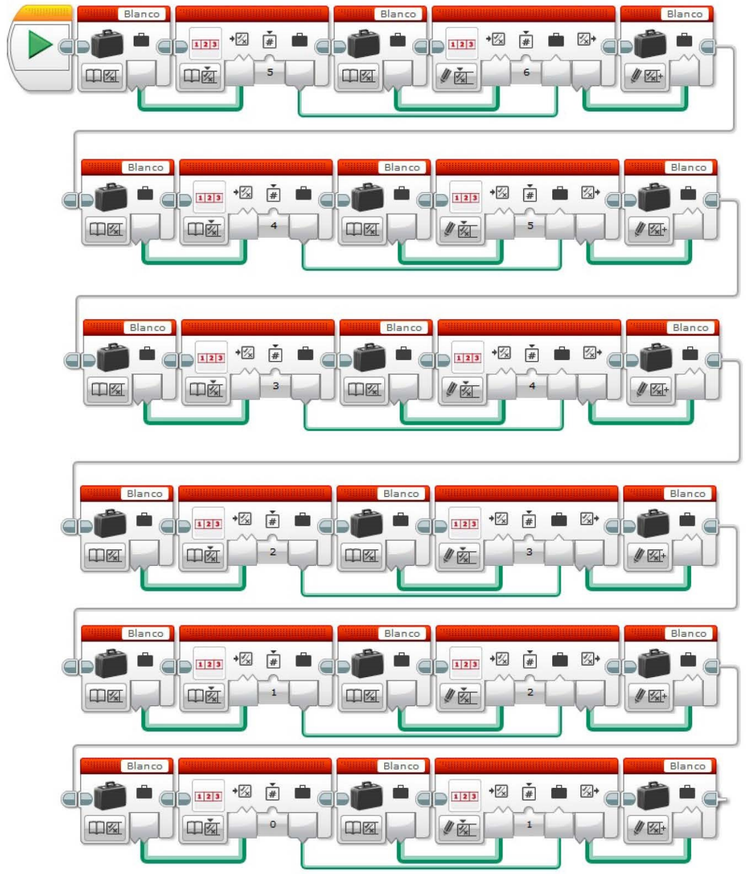






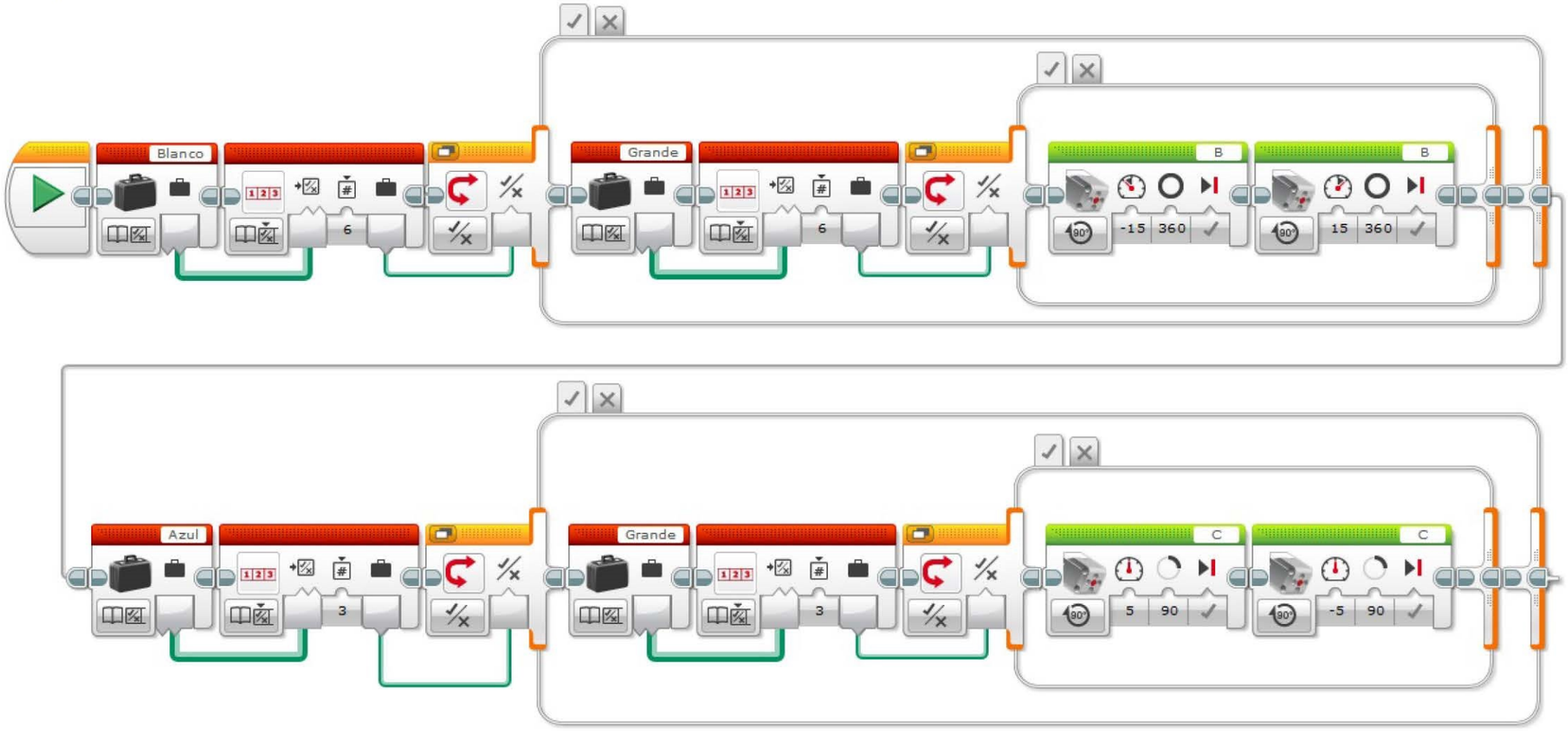


Bloque 'desplaza'



UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: CINTA CLASIFICADORA 1	FECHA: FEBRERO 2019 ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO FIRMA:	PLANO Nº: 08

Bloque 'Actuadores'



UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A164

TÍTULO DEL TFG:

LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO

TÍTULO DEL PLANO:

CINTA CLASIFICADORA 2

FECHA: FEBRERO 2019

ESCALA: Sin escala

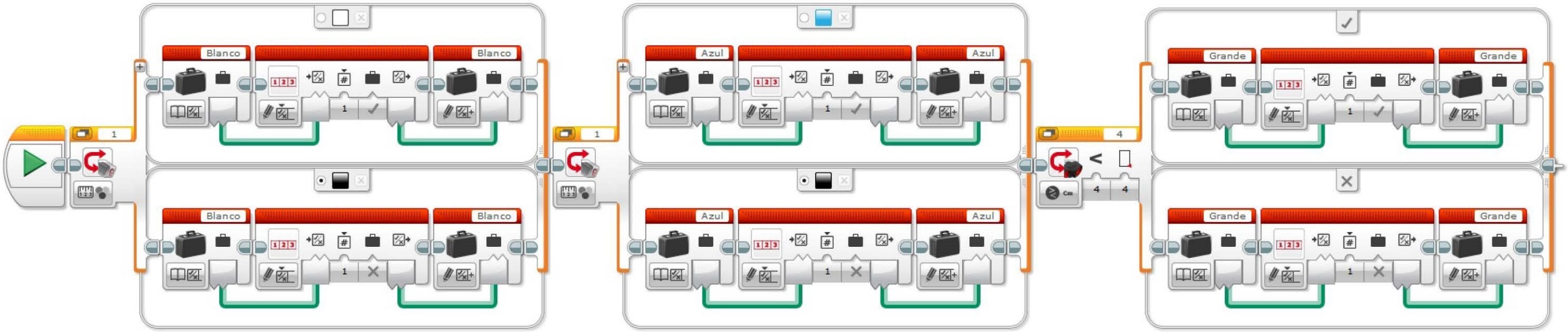
AUTOR:

DAVID GÓMEZ OCAMPO

FIRMA:

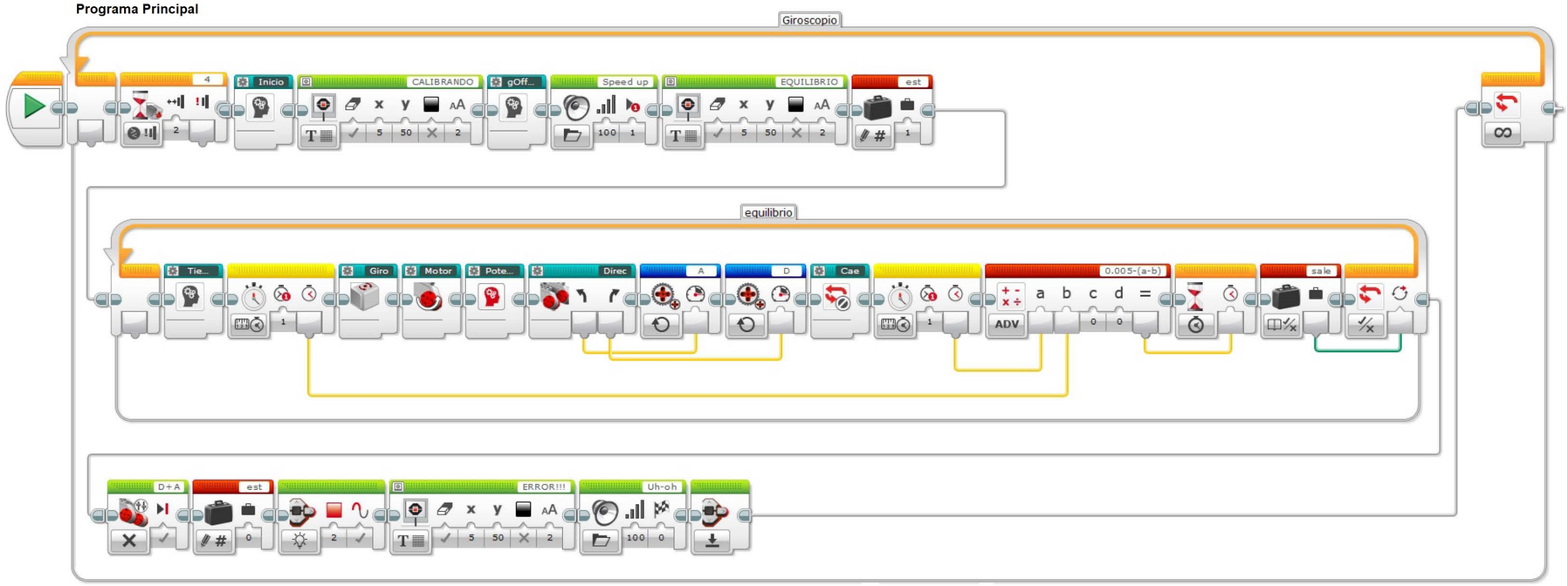
PLANO Nº: 09

Bloque 'Lee_sensores'

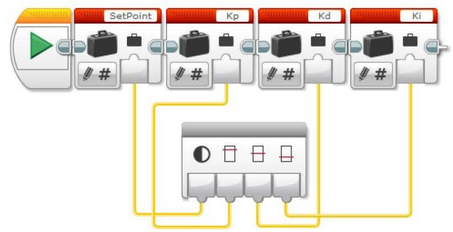


UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: CINTA CLASIFICADORA 3	FECHA: FEBRERO 2019
AUTOR: DAVID GÓMEZ OCAMPO — FIRMA:	ESCALA: Sin escala
	PLANO Nº: 10

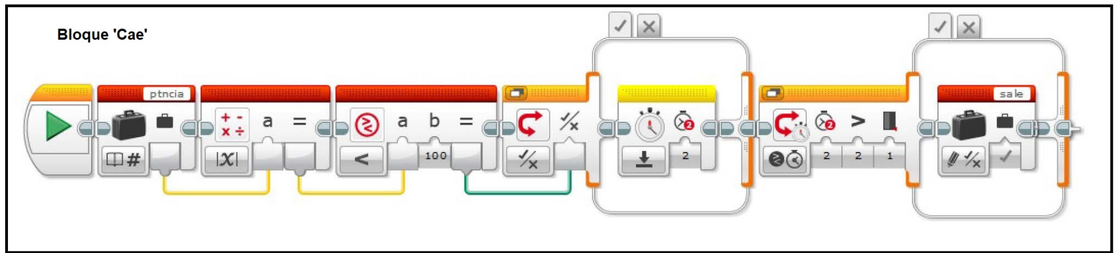
Programa Principal



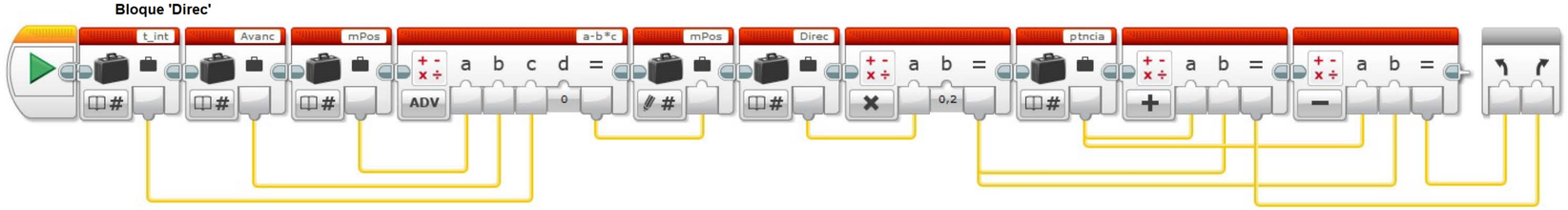
Bloque 'Conf'



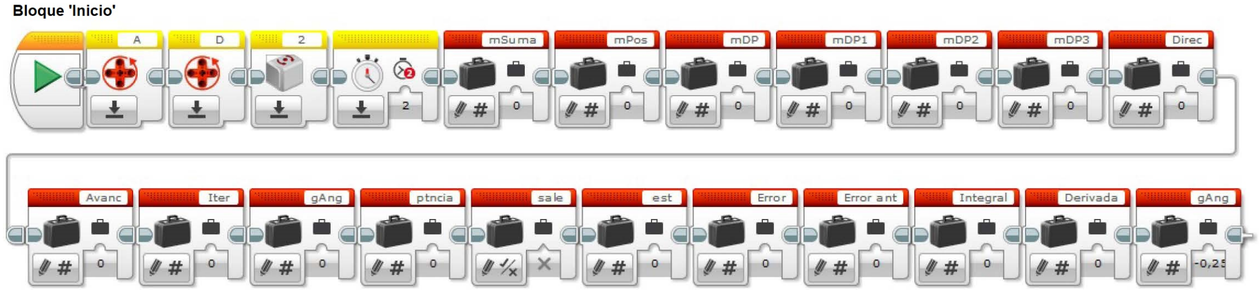
Bloque 'Cae'



Bloque 'Direc'

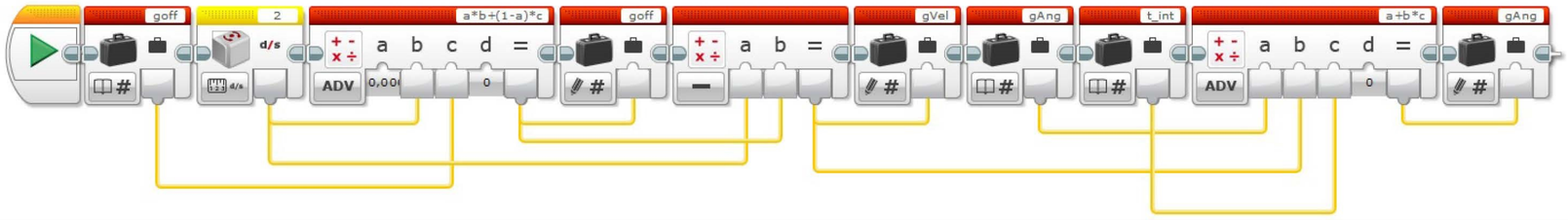


Bloque 'Inicio'

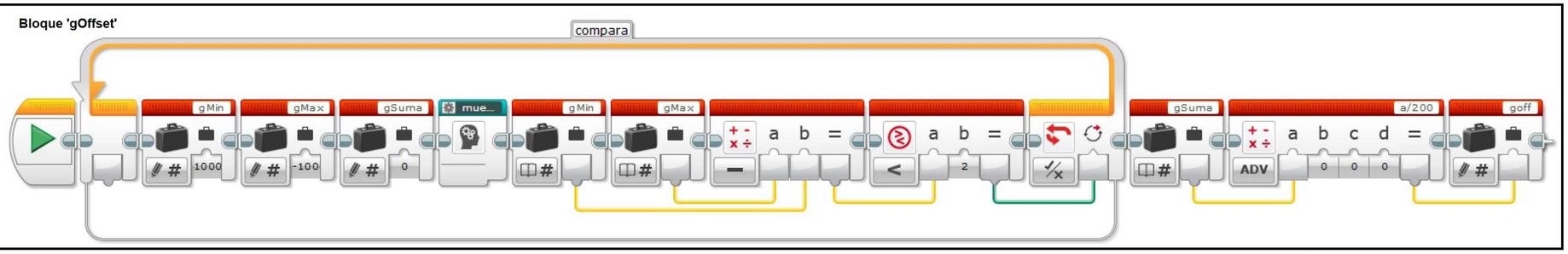


UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: ROBOT SEGWAY 1	FECHA: FEBRERO 2019
	ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO	FIRMA:
	PLANO Nº: 11

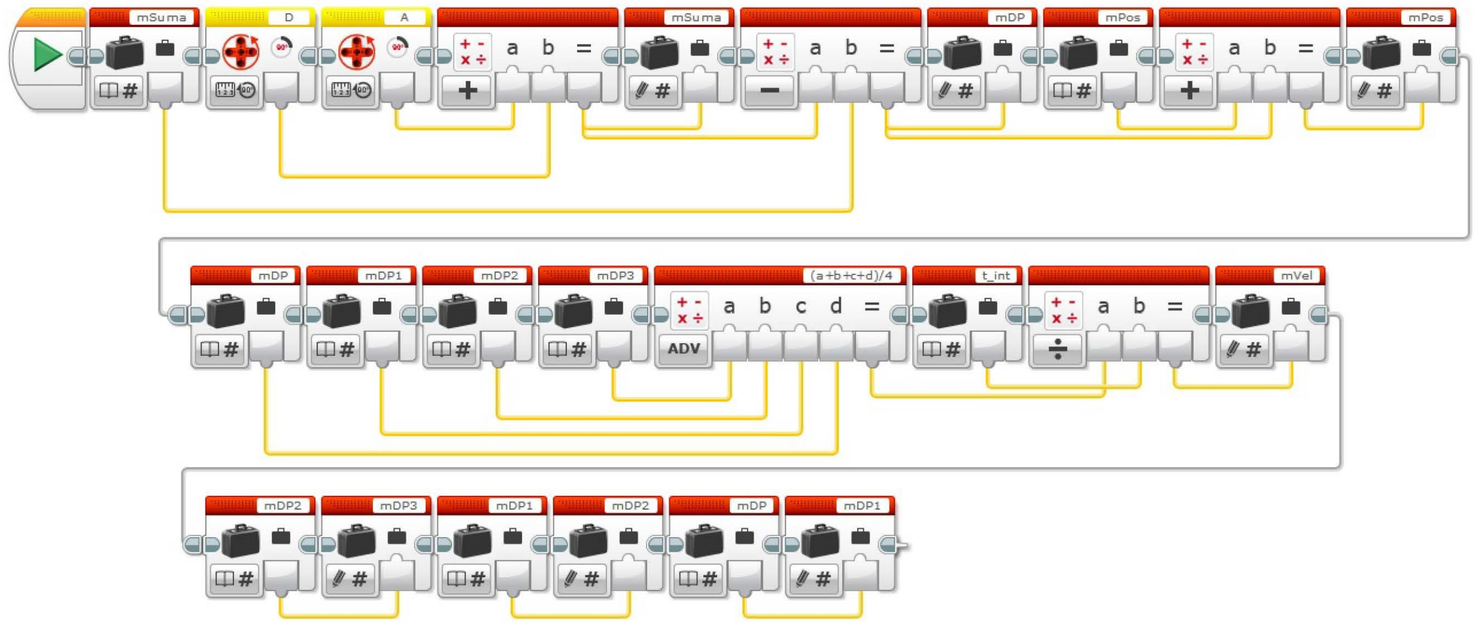
Bloque 'Giro'



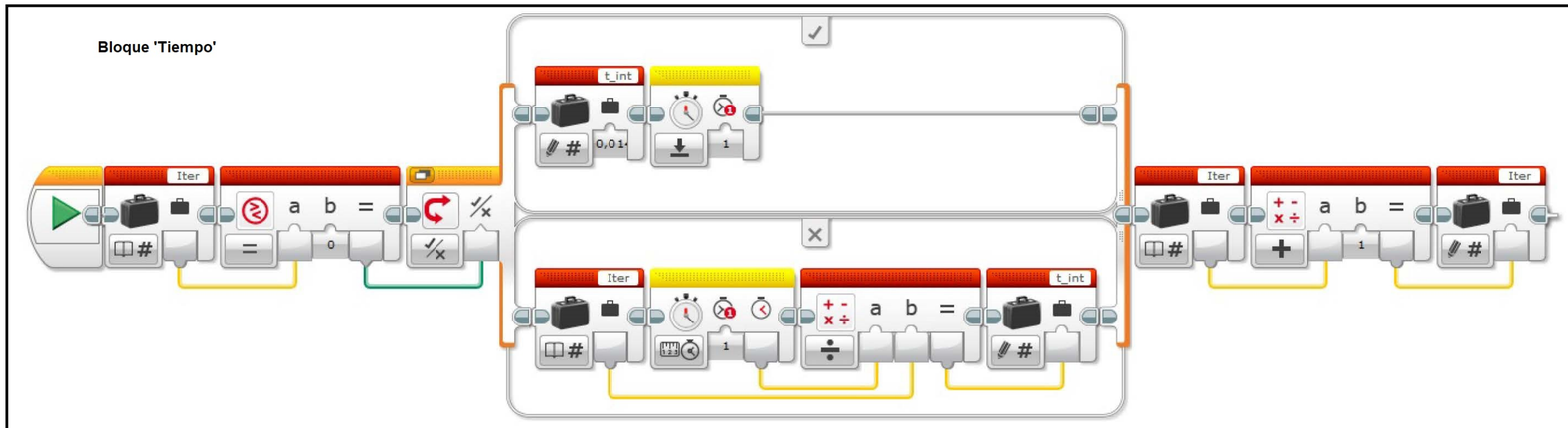
Bloque 'gOffset'



Bloque 'Motor'

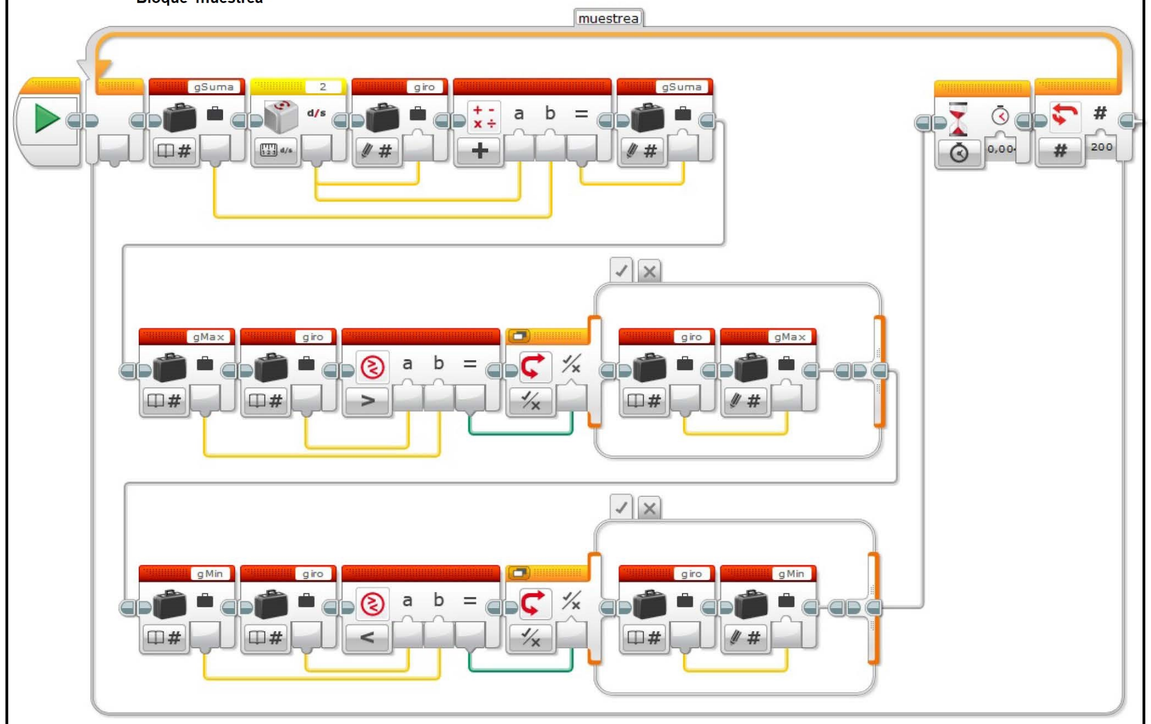


Bloque 'Tiempo'

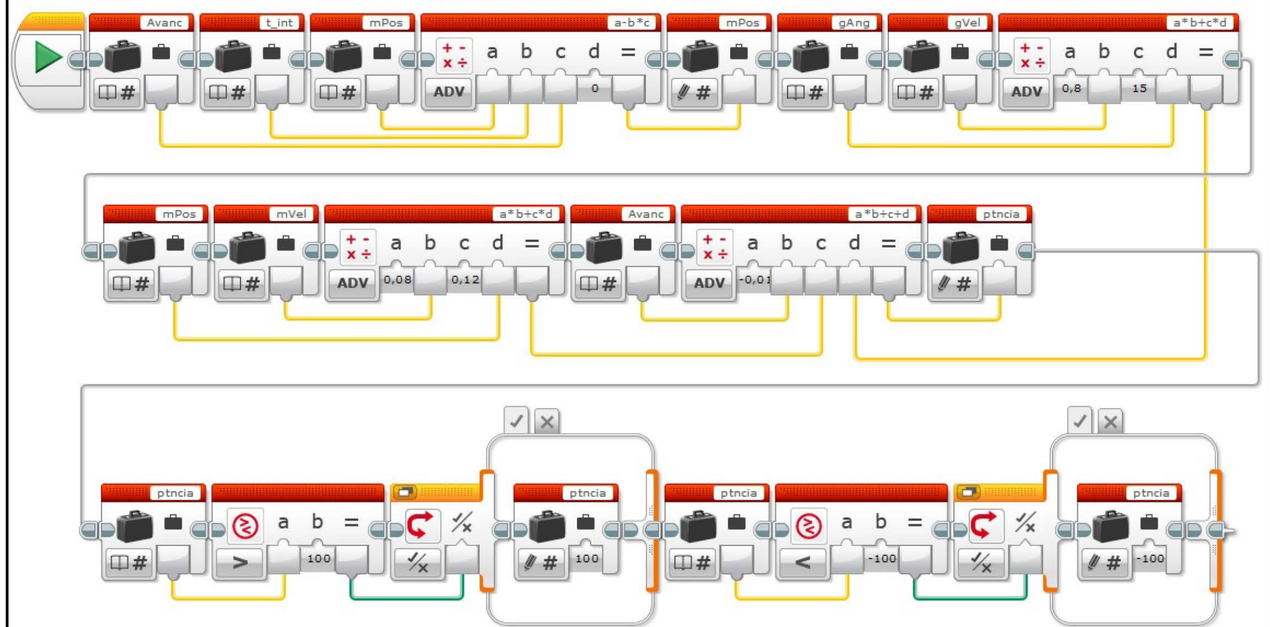


UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA		TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO		
TÍTULO DEL PLANO: **ROBOT SEGWAY 2**		FECHA: FEBRERO 2019 ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO	FIRMA:	PLANO Nº: 12

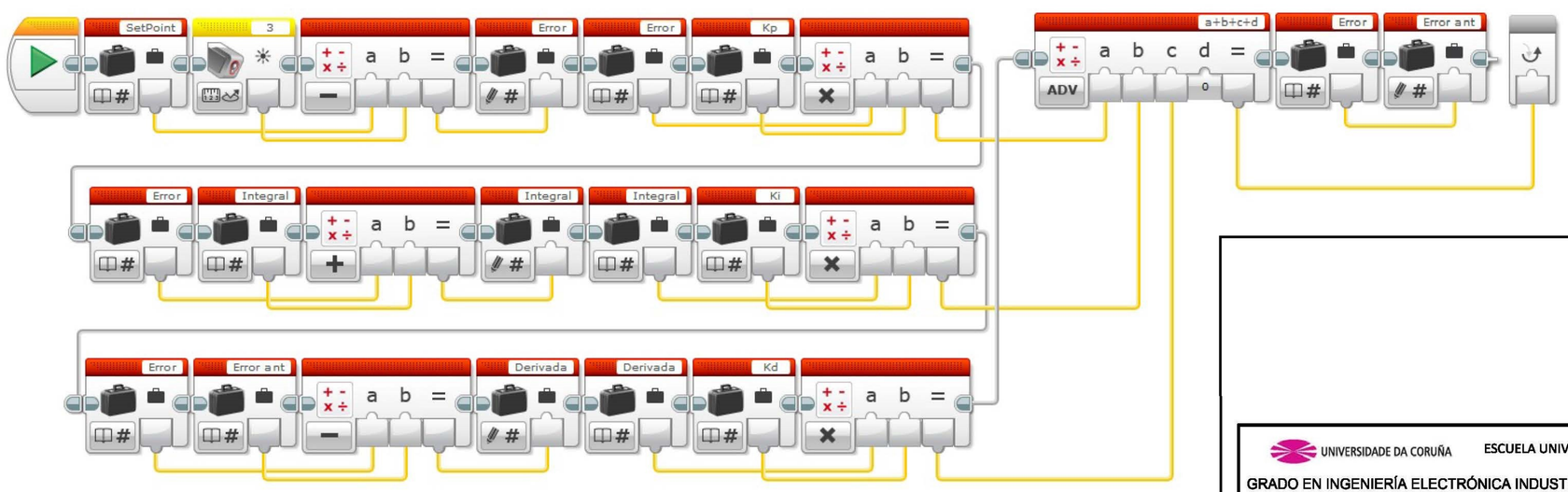
Bloque 'muestrea'



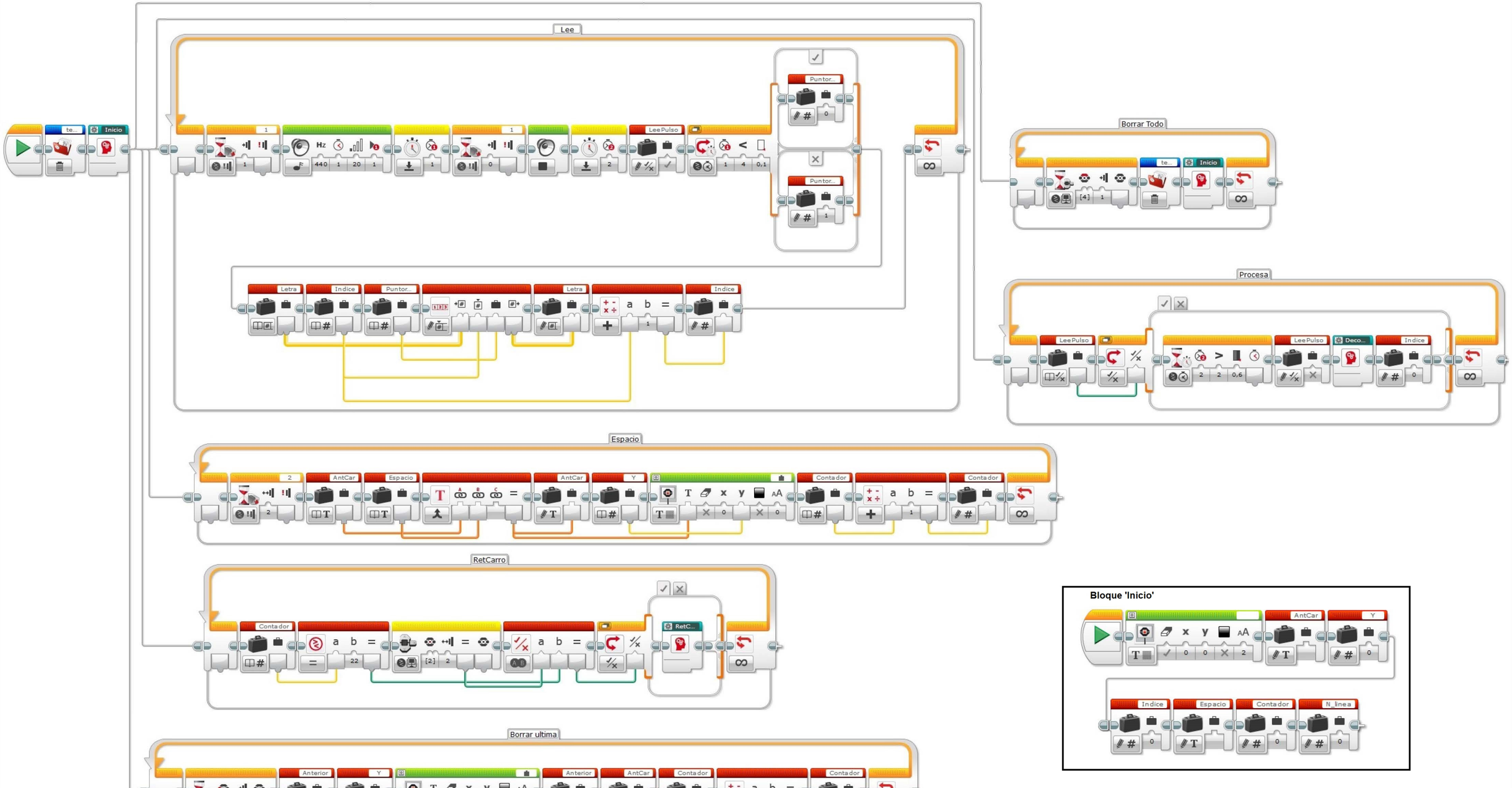
Bloque 'Potencia'



Bloque 'PID'

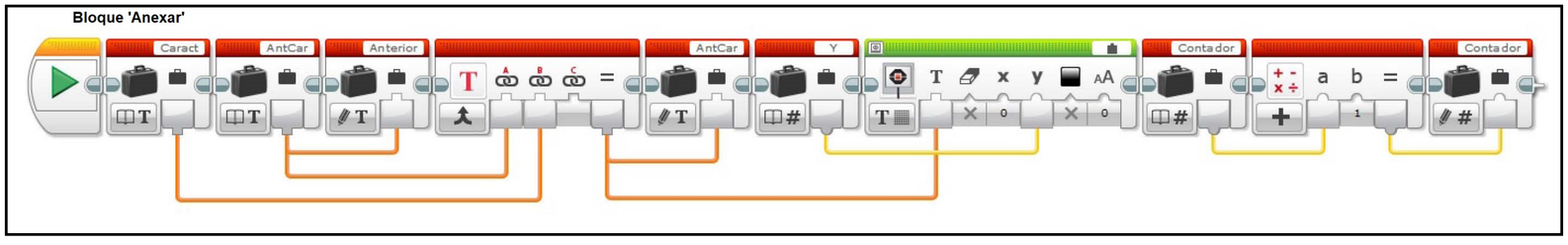


UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: ROBOT SEGWAY 3	FECHA: FEBRERO 2019 ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO — FIRMA:	PLANO Nº: 13



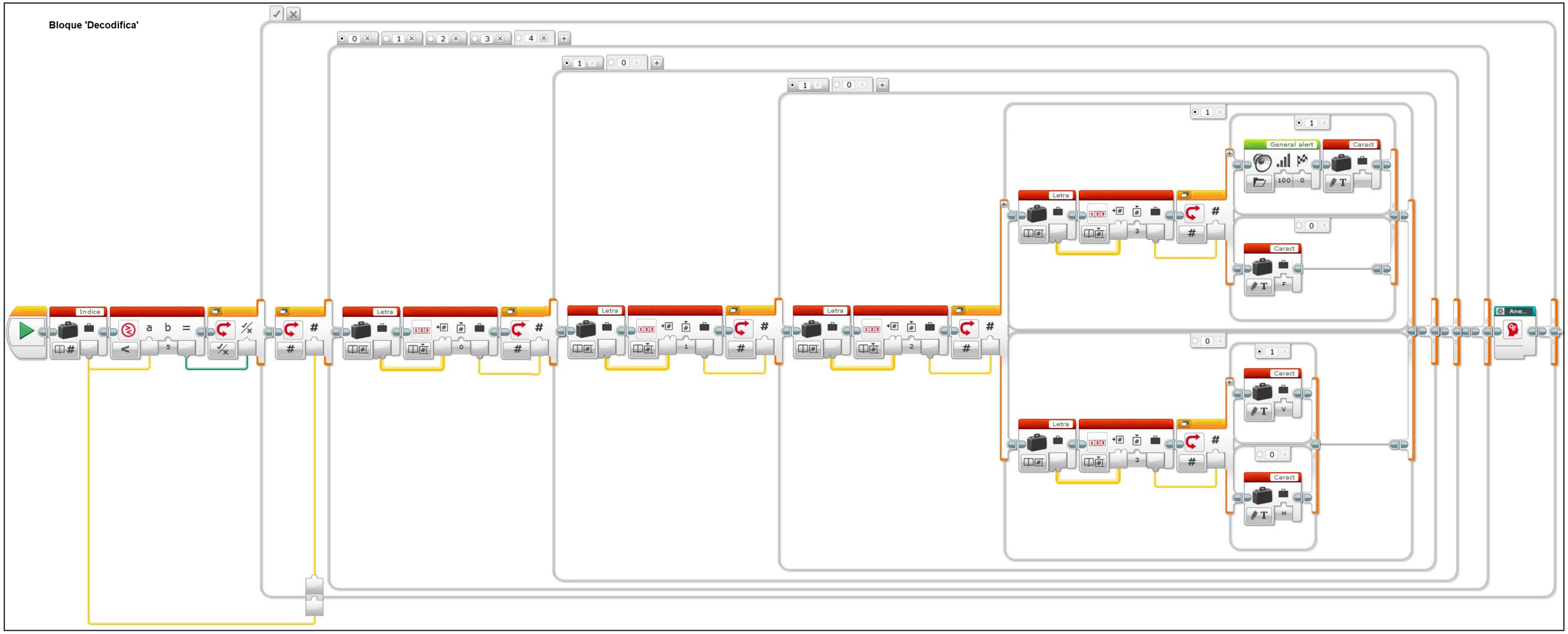
Bloque 'Inicio'

Bloque 'Anexar'

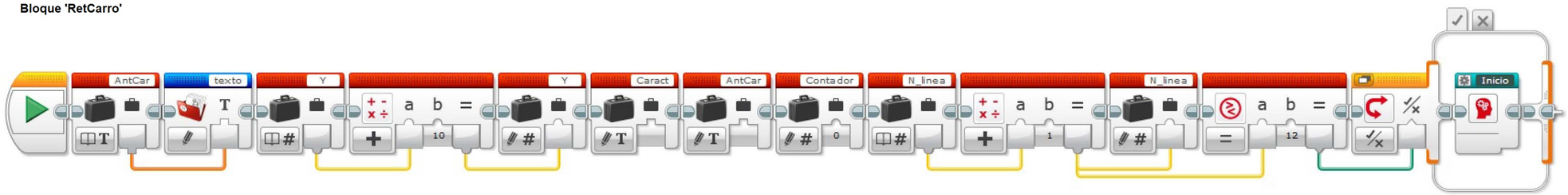


UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: TELÉGRAFO 1	FECHA: FEBRERO 2019 ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO FIRMA:	PLANO Nº: 14

Bloque 'Decodifica'

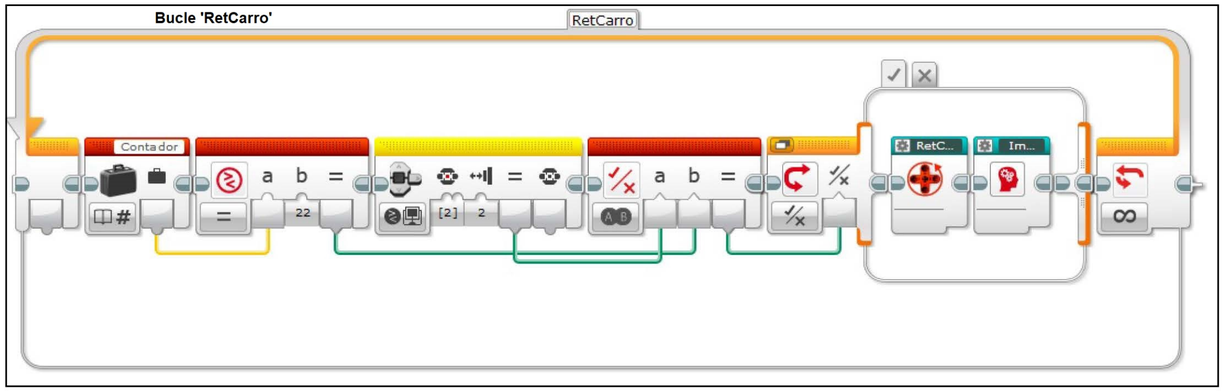


Bloque 'RetCarro'

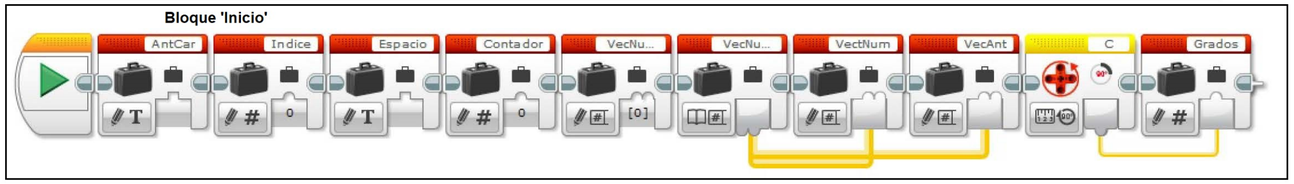


UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA		TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO		
TÍTULO DEL PLANO: TELÉGRAFO 2		FECHA: FEBRERO 2019 ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO	FIRMA:	PLANO Nº: 15

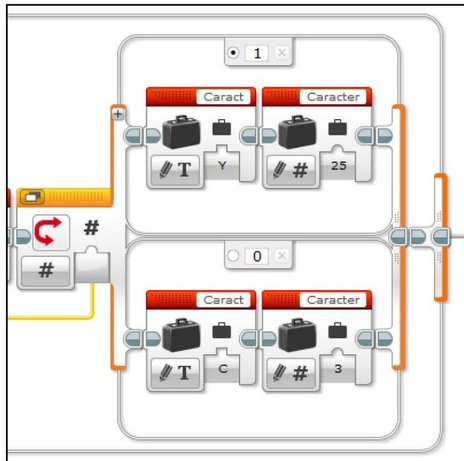
Bucle 'RetCarro'



Bloque 'Inicio'

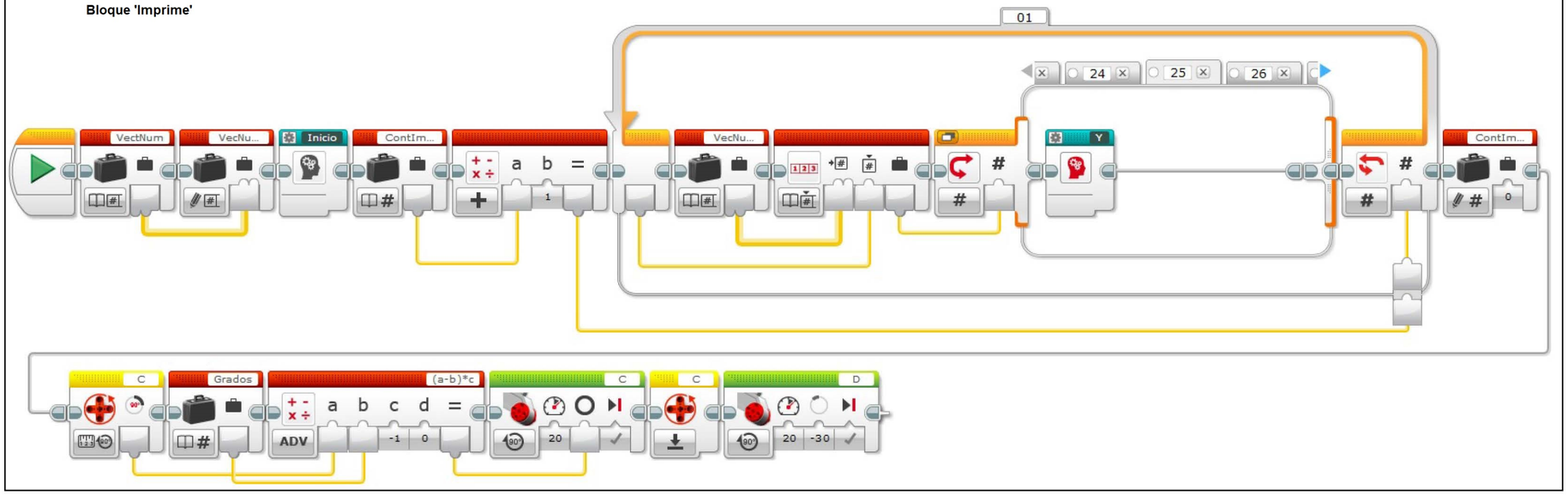


Bloque 'Decodifica' (Detalle)

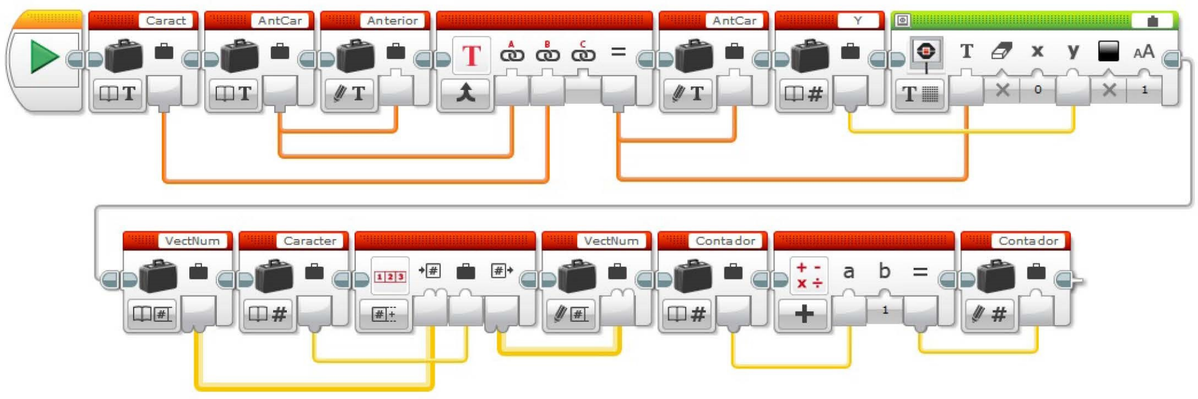


En el bloque 'Decodifica', respecto al ejemplo anterior (Telégrafo), además de la variable texto "Caract" se asocia una variable numérica "Caracter", ésta se concatenará en el vector numérico VecNum (Bloque 'Anexar'), este vector numérico se utilizará en el bloque 'Imprime'.

Bloque 'Imprime'



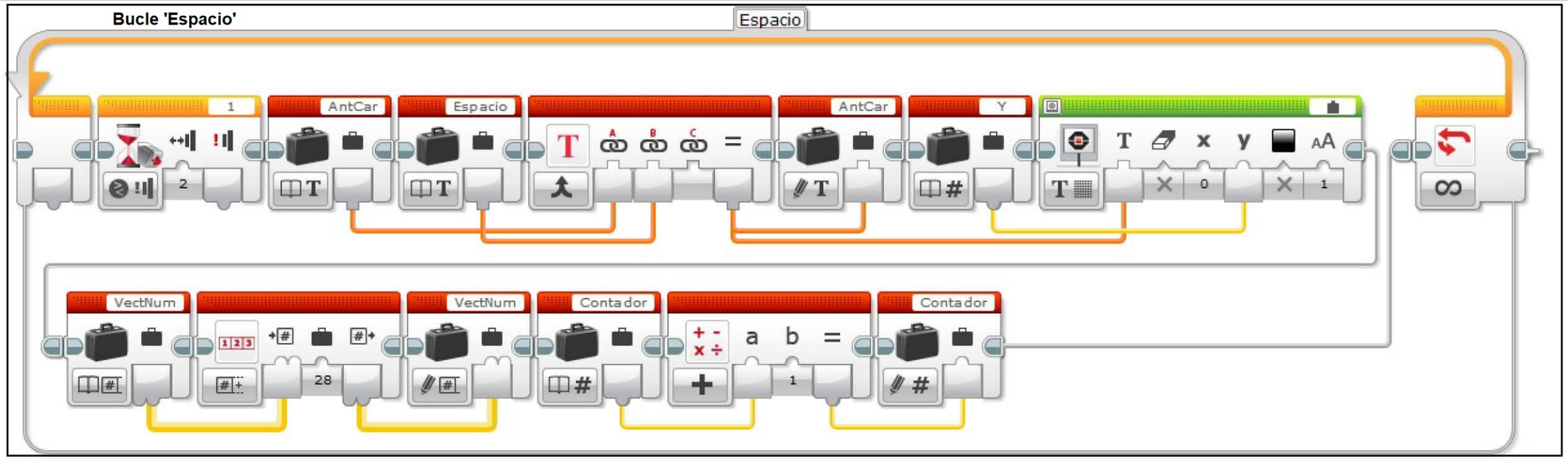
Bloque 'Anexar'



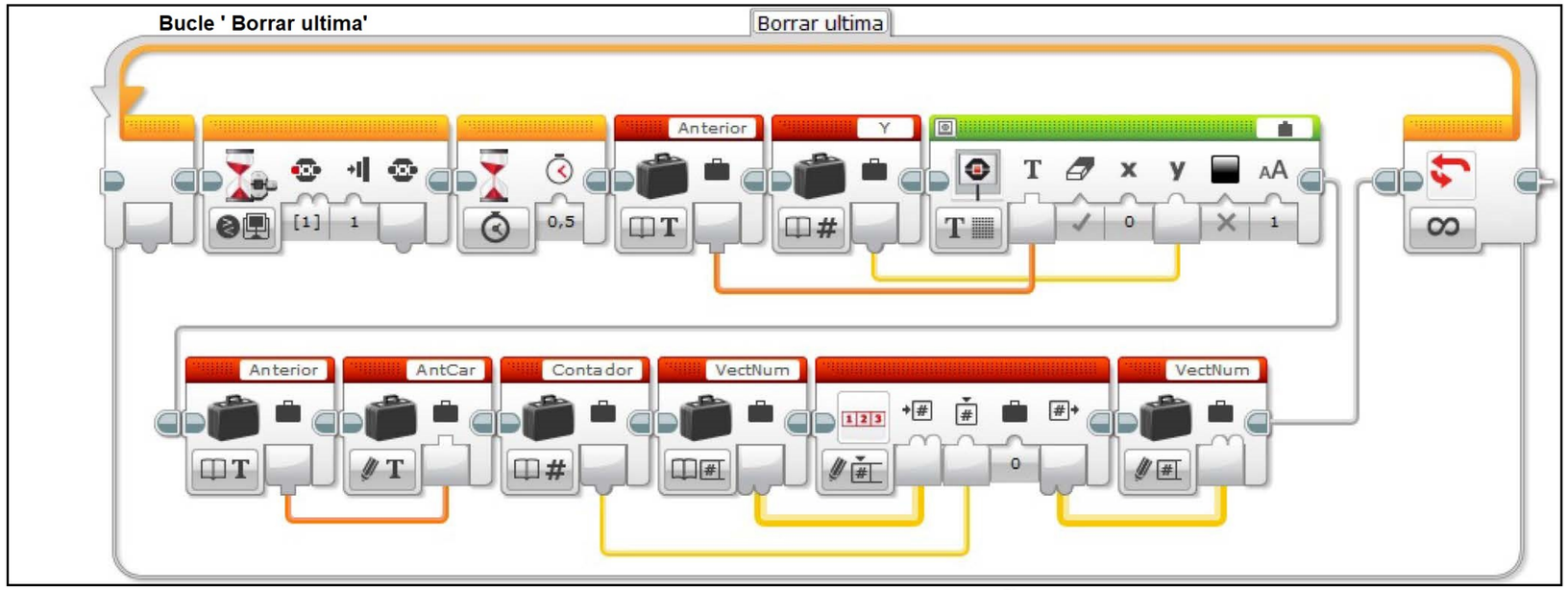
Nota: La estructura del programa es igual a la del ejemplo anterior (Telégrafo), solo se presentan los bucles que han sido modificados y los nuevos bloques añadidos sobre el ejemplo anterior.

UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: **TELÉGRAFO/IMPRESORA 1**	FECHA: FEBRERO 2019
	ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO — FIRMA:	PLANO Nº: 16

Bucle 'Espacio'



Bucle ' Borrar ultima'



Nota: La estructura del programa es igual a la del ejemplo anterior (Telégrafo), solo se presentan los bucles que han sido modificados.

UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A164
TÍTULO DEL TFG: LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO	
TÍTULO DEL PLANO: TELÉGRAFO/IMPRESORA 2	FECHA: FEBRERO 2019 ESCALA: Sin escala
AUTOR: DAVID GÓMEZ OCAMPO FIRMA:	PLANO Nº: 17

TÍTULO: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**

PLIEGO DE CONDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: FEBRERO DE 2020

AUTOR: EL ALUMNO

Fdo.: DAVID GÓMEZ OCAMPO

Índice del documento PLIEGO DE CONDICIONES

15 PLIEGO DE CONDICIONES

15.1. Requisitos del sistema

Las especificaciones mínimas de hardware de las que debemos disponer serán las siguientes:

Para Mindstorms EV3:

- Sistema operativo Microsoft Windows 10, Windows 7 o Windows Vista (soporta las versiones 32 y 64bit).
- Procesador Dual core, 2.0 Ghz o superior.
- 2GB de RAM o superior.
- 2GB de espacio disponible en disco.
- XGA (1024 x 768) o superior.
- 1 puerto USB disponible (para la conexión del bloque EV3)

Para LDD:

- Sistema operativo Microsoft Windows XP, Windows Vista, Windows 7, Windows 8 o Windows 10.
- CPU: 1 GHz o superior.
- Tarjeta gráfica: 128 MB (OpenGL 1.1 or superior compatible)
- RAM: 512 MB
- 1 GB de espacio disponible en disco.

15.2. Almacenaje

El material se almacenará en las cubetas que se incluyen en los kits, en un ambiente seco, en ausencia de polvo y temperaturas extremas. En el caso de usar pilas AAA como fuente de alimentación, se retirarán cuando el robot no vaya a ser usado en un espacio prolongado de tiempo.

TÍTULO: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**

MEDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: FEBRERO DE 2020

AUTOR: EL ALUMNO

Fdo.: DAVID GÓMEZ OCAMPO

Índice del documento MEDICIONES

16 ESTADO DE LAS MEDICIONES

17 Listado de materiales

Cantidad	Elemento	Ref
1	Set educativo LEGO® MINDSTORMS® EV3	45544
1	Set de expansión LEGO® MINDSTORMS® EV3	45560
1	Motor mediano adicional LEGO® MINDSTORMS® EV3	45503
1	Batería Recargable LEGO® DC EV3	45501
1	Cargador 10V DC LEGO®	45517

Tabla 17.1 – Listado de materiales

18 Distribución de horas

Actividad	Tiempo (h)
Montaje	35
Diseño 3D	10
Programación	30
Comprobaciones	10
Documentación	40

Tabla 18.1 – Distribución de horas

TÍTULO: **LIBRERÍA DE SOFTWARE DIDÁCTICO PARA UN ROBOT LEGO**

PRESUPUESTO

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: FEBRERO DE 2020

AUTOR: EL ALUMNO

Fdo.: DAVID GÓMEZ OCAMPO

Índice del documento PRESUPUESTO

19 PRECIOS UNITARIOS DE MATERIALES, MANO DE OBRA Y ELEMENTOS AUXILIARES

Elemento	Ref.	Precio (€/ud.)
Set educativo LEGO® MINDSTORMS® EV3	45544	432.95
Set de expansión LEGO® MINDSTORMS® EV3	45560	132.95
Motor mediano adicional LEGO® MINDSTORMS® EV3	45503	24.99
Batería Recargable LEGO® DC EV3	45501	98
Cargador 10V DC LEGO®	45517	30.95

Tabla 19.1 – Precio unitario de materiales

Elemento	Precio (€/h)
Mano de obra	30

Tabla 19.2 – Precio unitario de mano de obra

20 PRECIOS UNITARIOS DE LAS UNIDADES DE OBRA

21 PRESUPUESTO

Concepto	Precio (€)
Material	719.84
Mano de obra (horas)	3750
TOTAL	***4469.84***

Tabla 21.1 – Presupuesto total